Universitat Autònoma de Barcelona

POINT-BASED DATA COMPRESSION

Dion Eustathios Olivier Tzamarias

Universitat Autònoma de Barcelona

Departament d'Enginyeria de la Informació i de les Comunicacions

POINT-BASED DATA COMPRESSION

SUBMITTED TO UNIVERSITAT AUTÒNOMA DE BARCELONA
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

by Dion Eustathios Olivier Tzamarias

Bellaterra, June 2022

Supervisor:
Dr. Joan Serra-Sagristà
Dr. Ian Blanes

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Bellaterra, June 2022

Dr. Joan Serra-Sagristà
Dr. Ian Blanes

Committee:

Dr. Victor Francisco Sanchez Silva
Dr. Manuel Isaac Martinez Torres
Dr. Joan Bartrina Rapesta
Dr. Jordi Portell i de Mora (substitute)
Dr. Guillermo Navarro Arribas (substitute)

Contents

Chapter 1

Introduction

1.1 Introduction to the topic

The trend of the current technological era is the collection and accumulation of vast quantities and types of data. The purpose behind the acquisition of such a plethora of information can differ in many ways, yet frequent applications can be found in areas such as big data, machine learning, virtual/augmented reality, server optimization, etc. The nature of these data can vary in many ways, but a large subset of data-types can be categorized as point-based data. In other words, their component atoms are points, each carrying information relevant to its location and/or attribute, which is a value that can describe a color, a temperature, a flux etc.

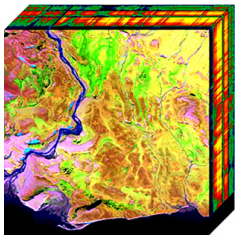
The points can be located in spaces of lower or higher dimensions, and they can be placed in strictly structured regular patterns or scattered irregularly in space. An example of regular point-based data are classic images, whose component atoms, i.e., pixels, are interpreted as color valued points placed on a regular grid. Usually, the pixel-points live in a 2D space such as the cases of natural or depth images. Diversely, hyperspectral images depict scenes across many bands of the electromagnetic spectrum, otherwise called spectral components, resulting in its constituent atoms being located in a 3D space.

Other point-based data, however, are not expressed by regular grid structures as they are represented by points, seemingly spread in random patterns. In this category belong such data groups that offer a digital three dimensional representation of objects, or en-

tire sceneries. Meshes and point clouds are such examples, whose points, apart from a geometric value, e.g., the point's coordinates, can also carry additional attributes such as color, reflectibility or norm. The non-uniform placement of these points is attributed to the acquisition techniques that sample irregularly a subject through the use of portable laser scanners or aerial photogrammetry systems. An illustration of point-based data, structured in a regular or irregular manner, can be found in Figure 1.1.

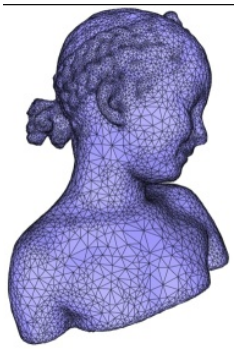
(a) Natural image: Baboon

(b) Natural image: Lena

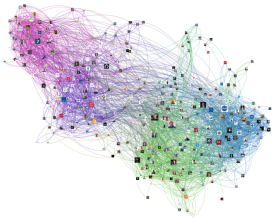


(c) Hyperspectral image

(d) Point cloud



(e) Mesh



(f) Social network

Figure 1.1: Depiction of point-based data whose atoms are placed in a regular-grid structure (a, b, and c), and whose atoms are spread irregularly (d, e, and f).

The dimensionality of the space in which point-based data live in is not restricted to the 3D space humans perceive. In the case of networks, for example, each point can represent a subject whose value is a feature vector and is interconnected with its neighbors through weighted edges. For example, social networks, used to simulate user interactions and trends, can place each point in a N-dimensional feature space, representing the interests, political views or economic status of a person.

Even though point-based data are a growing subset, historically it has been the focus of an ever evolving series of applications. Said data have been targets of successful and long lasting applications, as is the case of hyperspectral images, which are employed for soil type analysis, agriculture/forest monitoring and even to forecast geological events such as landslides. Other point-based data are found in the epicenter of many novel applications and one would expect their usage to grow over time. As an example, we refer to point clouds, which have been just recently employed in virtual/augmented reality, self driving cars, or heritage preservation applications, while 3D meshes are exceedingly used in the video game and movie industries.

The incessant accumulation of data demands large amounts of storage, which, apart from fabrication expenses, are the source of several additional difficulties. Maintenance of large storage facilities cumulates to huge costs, heavy storage hardware is scarce on board remote sensing satellites, and storage in small devices, such as phones or tablets, is limited on account of portability. However, the necessity for storage of point-based data can be alleviated through compression algorithms whose design depends on the type of information that the point carries.

When it comes to point-based data, classical compression techniques are not equally well suited for all its instances. For example, classical image compression algorithms are not suited for the compression of point cloud data. Furthermore, it is necessary for the compression scheme to adapt to the often irregular nature of point-based data such as in the case of epidemiological networks. This can be accomplished through the field of graph signal processing in which the attributes of point-based data are represented as graph signals, and whose inter-point correlation information is embedded in weighted graphs. Through the work of Ortega [1, 2, 3, 4, 5], who is considered a pioneer on the topic, many graph-based transforms have arisen and have also been used for the compression of graph signals.

In other point-based data, alternative methods are adopted, such as in the work of Garcia [6, 7, 8, 9] for the compression of the geometry information of point clouds.

1.2 Objectives

The motivation of this thesis is focused on the resolution of problems associated to the storage and transmission of point-based data, which is achieved through suitable compression algorithms. Therefore, our two main objectives are the research and design of novel algorithms for the compression of a) attribute and b) location information of point-based data. Regarding the first objective, the goal of our study had a more theoretical approach, exploring lossy as well as lossless designs, whereas the second objective is more practical in its nature, researching also fast and memory efficient methods.

In this dissertation, the means of achieving the attribute compression of point-based data are accomplished via the design and application of graph filterbank transforms. Point-based data are naturally represented by graph structures, as points form the nodes of a graph and similarities between them are embedded in the graph structure, in the form of weighted graph edges. The graph signal is referred to the collective attribute values of all points that reside on the graph. Utilizing the field of graph signal processing, tools such as wavelet filterbank transforms can be extended to the graph domain, resulting in techniques that accomplish the compression of graph signals [1, 10]. Although graph filterbanks can be employed for the compression of any graph signal, we have focused our research specifically on their application on hyperspectral and natural images.

When it comes to the compression of location information of point-based data, the chosen approaches are radically different in their design. In this thesis, the aforementioned aspect of point-based data encoding is explored through the lens of voxelized point cloud geometry compression (PCGC). A point cloud is composed by a collection of points in a 3D space forming a three dimensional representation of people, objects and even entire cities. A point cloud's geometry information refers to the precise coordinates of each point, which typically average to many hundreds of thousands, rendering geometry compression algorithms extremely important. Geometry compression methods are usually applied on geometrically quantized point clouds also known as voxelized. A voxel space is a regular

grid whose unitary cells are called voxels. The irregularly located points of a point cloud are represented by occupied voxels while the complementary empty space is formed by empty voxels. Thus, the goal of geometry compression algorithms (applied on voxelized point clouds) is to encode the precise coordinates of such occupied voxels.

1.3 Summary of Contributions

Our contribution to the field of point-based data compression extends to the compression of both the attribute and location of points. The attribute compression of points, which is more theoretical in nature, is researched by developing novel graph filterbank transforms for the compression of natural as well as hyperspectral images. We have developed the first lossy to lossless hyperspectral encoder based on graph filterbanks providing two discrete variants. Our experimental results show that the developed transforms outperform the widely used DWT in the lossy setting, while yielding an on-par performance on the lossless setting. Our subsequent theoretical work on the topic of graph filterbanks relates to previously open ended questions regarding the behavior of the well established biorthogonal graph filterbank transform known as the GraphBior [3]. Supported by recent developments in the field, we identify the reasons behind the drawbacks of the two original variations of GraphBior and propose a novel variation for the compression of natural images that balances the original drawbacks.

The compression of the points' location information is investigated by contributing to the field of point cloud geometry compression. Our initial work in this topic improved on the state of the art, resulting at the time in the most competitive lossless geometry compression scheme, in terms of rate. This study highlighted drawbacks of relevant methods and laid the foundations for our subsequent work. Along with state-of-the-art performance, our next publication brought drastic improvements in computational complexity and memory usage, both fundamental properties for a geometry compressions algorithm intended for real world use. Experimental results confirm that our most recent point cloud geometry compression algorithm surpasses MPEG's TMC13 standard in encoding speeds and compression performance.

To summarize, the subject of this dissertation is on on the topic of point-based data

compression, and our relevant work has been published in two conference papers and two journal papers. Our contribution to the field of point-based attribute compression involves hyperspectral and natural image compression through novel graph filterbank transforms. Regarding our contribution to point-based location compression, our two publications involve the geometry compression of point cloud data. The following list summarizes our relevant publications in chronological order.

- D.E.O. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagristà. "Compression of hyperspectral scenes through integer-to-integer spectral graph transforms." Remote Sensing 11, no. 19 (2019): 2290, JCR impact factor: 4.848.

- D.E.O. Tzamarias, E. Pavez, B. Girault, A. Ortega, I. Blanes, and J. Serra-Sagristà. "Orthogonality and Zero DC Tradeoffs in Biorthogonal Graph Filterbanks." In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5509-5513. IEEE, 2021, CORE quality factor: B.

- D.E.O. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagristà. "Compression of point cloud geometry through a single projection." In 2021 Data Compression Conference (DCC), pp. 63-72. IEEE, 2021, CORE quality factor: A+.

- D.E.O. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagristà. "Fast run-length compression of point cloud geometry." In 2022 IEEE Transactions on Image Processing, vol. XX, pp. XXXX, IEEE, 2022, JCR impact factor: 10.856.

Chapter 2

Compression of hyperspectral scenes through integer-to-integer spectral graph transforms

JCR impact factor: 4.848

remote sensing

MDPI

Article

Compression of Hyperspectral Scenes through Integer-to-Integer Spectral Graph Transforms †

Dion Eustathios Olivier Tzamarias *, Kevin Chow, Ian Blanes and Joan Serra-Sagristà

Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Campus UAB, 08193 Cerdanyola del Vallès, Barcelona, Spain; kevin.chow@uab.cat (K.C.); ian.blanes@uab.cat (I.B.); joan.serra@uab.cat (J.S.-S.)

* Correspondence: dion.tzamarias@uab.cat

† This paper is an extended version of our paper published in in the Proceedings of the 6th ESA/CNES International Workshop on On-Board Payload Data Compression (OBPDC), Matera, Italy, 20–21 September 2018, titled *Compression Of hyperspectral images with graph wavelets*.

Received: 30 August 2019; Accepted: 26 September 2019; Published: 30 September 2019

Abstract: Hyperspectral images are depictions of scenes represented across many bands of the electromagnetic spectrum. The large size of these images as well as their unique structure requires the need for specialized data compression algorithms. The redundancies found between consecutive spectral components and within components themselves favor algorithms that exploit their particular structure. One novel technique with applications to hyperspectral compression is the use of spectral graph filterbanks such as the GraphBior transform, that leads to competitive results. Such existing graph based filterbank transforms do not yield integer coefficients, making them appropriate only for lossy image compression schemes. We propose here two integer-to-integer transforms that are used in the biorthogonal graph filterbanks for the purpose of the lossless compression of hyperspectral scenes. Firstly, by applying a Triangular Elementary Rectangular Matrix decomposition on GraphBior filters and secondly by adding rounding operations to the spectral graph lifting filters. We examine the merit of our contribution by testing its performance as a spatial transform on a corpus of hyperspectral images; and share our findings through a report and analysis of our results.

Keywords: hyperspectral image coding; graph filterbanks; integer-to-integer transforms; graph signal processing

1. Introduction

Hyperspectral images are representations of scenes across many bands of the electromagnetic spectrum (otherwise called spectral components). These images have been used to classify areas of a landscape [1], identify clouds [2] or seeds located on the Earth's surface [3] and even forecast geological events such as landslides [4]. Due to their large size, these images require the use of efficient compression techniques in the form of specialized image transforms. The above mentioned transforms can often be computationally costly but one could strive towards real-time compression of remotely-sensed hyperspectral images, when processed accordingly through on-ground, high-performance parallel computing facilities.

Transform-based lossy compression techniques used on hyperspectral images are commonly related to one of two families. The first group of algorithms are based on the Discrete Wavelet transform (DWT) [5–8]. On one hand the advantage of these transforms manifests through their low computational complexity, though they are not adaptive to the input signal. On the other hand, algorithms such as those in References [9–12] are based on the Karhunen-Loeve transform (KLT), which is adaptive at the expense of a high computational complexity. Meanwhile the techniques used for the

lossless compression of hyperspectral images are generally based on a predictive coding model [13,14] though lossy predictive techniques [15] have the benefit of a lower computational complexity. Some lossless predictive techniques [16,17] are implemented in the CCSDS-123.0-B-2 [18] standard that also supports near-lossless compression and whose parameter tuning is explored in Reference [19].

This paper exploits a recently emerged family of wavelet transforms which are based on graph signal processing, a field in which data structures are represented as signals lying on the nodes of graphs while weighted graph edges denote the degree of similarity between nodes [20,21]. Graph wavelet transforms are derived from the graph structures, which in turn, one can also construct from images. Applying such transforms on hyperspectral images has been shown to lead to competitive compression results [22].

The strength of graph wavelet transforms in the field of image compression stems from the Graph Fourier Transform (GFT), an equivalent to the classical Fourier transform that is formed by the eigenvectors of the graph Laplacian, a matrix constructed from the values of the weighted edges of the graph. Just like the KLT, the GFT is an adaptive transform but instead of following a statistical approach, it attempts to encode image structures that are embedded in the graph edges. Thus, an important attribute of the GFT is related to its flexibility, since one can decide on the degree of accuracy with which image structures are represented on the graph [23]. It has also been shown [24] that not only does the GFT approximate the KLT for a piece-wise first-order autoregressive process but also that the GFT optimally decorrelates images following a Gauss-Markov random field model [25]. Consequently, graph wavelet transforms combine the GFT with a multiresolution analysis, resulting in a powerful tool for image compression.

A brief review of graph wavelet transforms follows. One can organize these transforms into two general groups—vertex domain and spectral domain designs. The former are based on spatial features of the graph such as the degree of connectivity between nodes. These vertex domain designs, though, lack spectral localization. While the energy of the resulting transformed signal is not concentrated around central graph frequencies, the vertex domain designs are described by a perfect localization on the vertex domain, meaning that one can define the number of nodes that will update the value of a vertex after the transform. One specific vertex domain design is the lifting-based graph wavelets [26] that use distinct groups of nodes to compute the update and detail wavelet coefficients.

Transforms that belong in the latter category of spectral designs exploit characteristics of the graph spectrum (not to be confused with the spectrum of hyperspectral images) in the form of the eigenvectors and the eigenvalues of the graph. One key feature of these designs is good spectral as well as vertex domain localization. One of the first spectral graph wavelet transforms are the diffusion wavelets [27] as well as the spectral graph wavelets [28]. While these spectral graph transforms are over-complete (the number of wavelet coefficients surpass the number of signal samples), this problem is solved by the two-channel graph wavelet filterbanks [29] that uses quadrature mirror filters (QMF) on bipartite graphs. These filters are not compactly supported and produce transforms that are not well localized on the vertex domain. The next iteration of such transforms are the biorthogonal graph wavelet filterbanks (introducing the GraphBior transform) [30] which are compactly supported as well as critically sampled. Since then several improvements and variations of the biorthogonal graph filterbanks have been proposed by including spectral sampling [31] or by introducing the M-channel graph wavelet filterbanks that can be implemented on large sparse graphs [32]. One of these variations uses polyphase transform matrices [33], proposing graph lifting structures [34] in the spectral domain for biorthogonal graph filterbanks.

One major issue of the filters developed for biorthogonal graph filterbanks, regarding image compression, is that they are only suited for lossy compression schemes. This drawback originates from the fact that the graph wavelet coefficients arising from these transforms are not integer. Thus a necessary quantization step is required rendering such compression schemes lossy. Furthermore, the use of spectral graph transforms for the compression of hyperspectral images spawns further

difficulties—a huge amount of side information is required, due to the necessity to transmit the graph structure to the decoder.

In this paper, we provide transforms that allow us to construct a lossy-to-lossless compression scheme for hyperspectral images, using biorthogonal graph filterbanks. We developed two integer graph-transforms, both suitable for the biorthogonal graph filterbanks.

Our first approach is to modify the filtering process of GraphBior so that the resulting analysis wavelet coefficients are integer. We solve this problem by computing the Triangular Elementary Rectangular Matrix (TERM) [35] decomposition of the spectral GraphBior filter. At first glance this solutions seems promising but due to the high complexity of the TERM factorization process and the large size of the GraphBior filters, one could argue otherwise. Thus we partition each GraphBior filter in tiles and compute the TERM factorization of each tile in parallel. This process dramatically decreases the time complexity of the TERM factorization of GraphBior.

Our second integer spectral graph transform is achieved by introducing rounding operations within the spectral graph lifting structures proposed in Reference [34]. Hence, we transform it into an integer-to-integer graph wavelet transform.

Using our proposed transforms mentioned above, we design a lossy-to-lossless extension to the scheme developed for the lossy compression of hyperspectral images through GraphBior in Reference [22]. The compression scheme published in Reference [22] tackles the issue of transmitting the graph structure to the decoder by assembling consecutive hyperspectral components into packets called band groups.

In this paper we evaluate and analyze the performance of our lossy-to-lossless compression scheme on a variety of hyperspectral images. We then explore the use of our graph transforms for hyperspectral images. Additional, we explore the effect that different parameters have on our compression scheme. Such parameters are the size of the tiles as well as the size of band groups. Experimental results are provided comparing the performance of the proposed techniques for several images from the CCSDS MHDC corpus of hyperspectral images [36].

This paper is structured as follows. First, in Section 2, we present a brief overview of graph signal processing and biorthogonal graph filterbanks. Then, the derivation of the TERM decomposition of the GraphBior transform, the introduction of the integer spectral graph lifting transform, as well as the adopted coding strategy are discussed in Section 3. In Section 4 we detail the setting of our experiments and showcase our results. Our conclusions are stated in Section 5.

2. Graphs and Biorthogonal Graph Wavelets

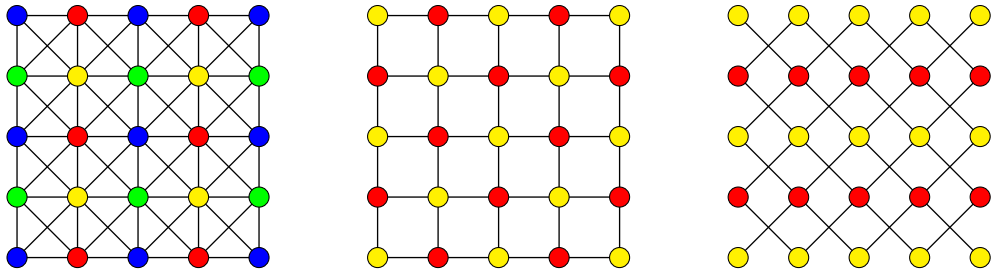
This section provides a brief introduction on graph signal processing and GraphBior. A graph $G = (V, E)$ is composed by a set of nodes V that are linked to each other by a set of edges E. There is a large variety of graphs, each with its own special properties. One relevant example is the bipartite graph, whose nodes can be arranged in 2 subsets such that there are no edges connecting nodes of the same subset. In other words, one needs only two colors in order to color the nodes of the graph in a way that no two nodes of the same color are connected through an edge. Two examples of bipartite graphs can be seen in Figure 1b,c.

The edges of a graph can be weighted, in a way that nodes can be connected strongly or weakly by a non discrete measure. The adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$, of a graph, is the symmetric matrix that describes the strength of all the possible connections between all nodes of a graph. We establish matrix A in such a way that any of its elements located in row i and column j is a real number $a_{ij} \in [0, 1]$ representing the weight of the edge between node i and j. The stronger the connection between two nodes, the higher the value of the weight, with the value 1 describing an edge of the highest strength. Conversely, weaker connections are represented by lower adjacency values such that a weight value equal to 0 represents a non existing edge between nodes. Using the Adjacency matrix one can compute the normalized Laplacian matrix that is defined as $\widetilde{L} = I - D^{-1/2}AD^{-1/2}$, where D is the diagonal degree matrix whose ith element is equal to the sum of the elements of A situated on its ith row.

In graph signal processing the normalized Laplacian matrix is of vital importance since it embeds within its structure the spectral information of the graph. Due to its symmetric nature, the eigenvectors of $\widetilde{L}$ create a complete orthogonal basis that can describe any vector in $\mathbb{R}^{|V|}$ as a linear combination of its basis. Additionally the eigenvalues of $\widetilde{L}$ are known as the spectrum of the graph and portray a notion of frequency.

We can also assign a real value to each node of the graph such that node i has a value of f_i. A graph signal $f \in \mathbb{R}^{|V|}$ is expressed mathematically as a vector and represents the collective values of the nodes. A graph signal of low frequency is expected to vary slowly, meaning that strongly connected nodes will support very similar graph signal values, while a high frequency signal is expected to assign very dissimilar values to strongly connected nodes. In other words a low frequency signal reflects the natural connectivity characteristics of the nodes of the graph whereas a high frequency signal goes against the connectivities imposed by the values of the graph edges.

The GraphBior transform, just as classical wavelet filterbanks, filters in the analysis step a signal into low pass and high pass signals that are later downsampled in order to decrease the number of graph wavelet coefficients. In order to guarantee reversibility of the transform, GraphBior exploits a particular characteristic of the spectrum of bipartite graphs (called spectral folding) [30]. Thus an arbitrary graph first needs to be decomposed into a series of bipartite subgraphs. These subgraphs share the same set of nodes as the original graph but their sets of edges do not intersect. The GraphBior filterbanks then makes use of these graphs by creating low and high pass filters.



(**a**) General graph (**b**) First bipartite component (**c**) Second bipartite component

Figure 1. A graph (**a**) is decomposed into bipartite components (**b**) and (**c**). Node colors are the result of the graph coloring process, required for the graph bipartition.

3. Lossy-to-Lossless Graph Wavelet Filterbanks

Our work in this paper strives towards a lossy-to-lossless extension of Reference [22] for the compression of hyperspectral images using the GraphBior filterbanks. In this following section we go through the compression scheme that is employed, which follows the one introduced in Reference [22]. Then we propose an integer-to-integer version of the GraphBior filters, by applying a TERM decomposition, as well as tiling to decrease the time complexity of the TERM factorization. Additionally, we propose a second integer-to-integer graph transform suited for the biorthogonal graph filterbanks by modifying the spectral graph lifting structures in Reference [34].

3.1. Compression Scheme

Our compression scheme consists of 3 modules—one whose purpose is to calculate the biorthogonal graph filterbank transform, an encoder and a decoder.

The first module is where a single hyperspectral component is used for the construction of a biorthogonal graph filterbank transform and follows the scheme introduced in Reference [22]. The component is first mapped into a graph, then decomposed into a series of bipartite graphs which will finally be used to create the graph transform. We design a graph G by representing the pixels of a component as graph nodes and connect neighboring nodes with vertical, horizontal and diagonal edges resulting in a 8-regular graph just as the one shown in Figure 1a. Edges between two connected

nodes are strong if the luminance values of these two pixels are similar and low when they are not. Specifically a graph edge a_{ij} is calculated through the Gaussian kernel $a_{ij} = e^{-\frac{(f_i - f_j)^2}{\sigma}}$ where f_i and f_j are the values of pixels i and j and σ is a scaling factor. Once the graph $G = (V, E)$ has been constructed, it is decomposed into a series of n bipartite subgraphs $B_i = (V_i, E_i)$ with $i = 1, \ldots, n$. The bipartition should be done in a way where each bipartite subgraph has the same vertex set as the original graph, whereas the edges of E are distributed among the subgraphs B_i in such a way that no single edge of the original graph is present in more than one bipartite subgraphs. Thus, the union of all the sets of edges from all the bipartite graphs is equal to the set of edges of the original graph. In other words, $V_i = V$, $\cup_i E_i = E$ and $E_i \cap E_j = \emptyset$ for $i \neq j$. To decompose a graph into a series of bipartite subgraphs, we utilize Harary's decomposition [37]. Other decomposition techniques are found in References [38–40]. In the case of the 8-regular graph, shown in Figure 1a, the bipartition resulting from Harary's algorithm is very intuitive and is not computationally intensive. By discarding all diagonal edges from the graph in Figure 1a, we construct the first bipartite graph, shown in Figure 1b and by discarding all horizontal and vertical edges from Figure 1a, we construct the second bipartite graph, shown in Figure 1c. Hence, an 8-regular graph G, constructed out of a single component, is decomposed into two bipartite graphs that are utilized for the computation of a biorthogonal graph filterbank transform, just as shown in Figure 2.

It is necessary, for the reversibility of the transform, that the encoder and the decoder have both access to the same graph structure. This ensures that both modules construct the analysis and synthesis filters out of the same graph. The first module is designed in a way that circumvents the transmission of the graph structure to the decoder without the need of side information. This is done by only using decoded components for the calculation of the graph wavelet transforms. Specifically, components are bundled into packets of consecutive components called band groups. Each band group should preferably consist of the same number, ω, of components. Once the decoder has decompressed a band group we extract the last of its components. Using that component we create the graph G and thus the graph wavelet transform. This process can be executed in the decoder and the encoder simultaneously. Once the graph wavelet transform has been learned, it is then applied to the next band group, in the analysis step within the encoder as well as in the synthesis step at the decoder. A schematic representing the computation and usage of the graph filterbank transform is depicted in Figure 3a.

The next module is the encoder, whose first operation upon receiving an image is to partition all but the first component into band groups. The first component is then compressed in a lossless manner by a spatial DWT and transmitted to the decoder. Thus the first component can be used to create the biorthogonal graph filterbank transform for the first band group. Remaining components are partitioned in band groups and encoded sequentially. The next step involves an optional RKLT or DWT transform across the spectral direction of the band group. This step is followed by the application of a spatial graph wavelet transform and the resulting graph wavelet coefficients are then quantized and coded by a plain entropy encoder. Accordingly, the module of the decoder is composed by an entropy decoder, a dequantizer, the synthesis process of the biorthogonal graph filterbank and the optional inverse spectral transform. The encoder and decoder modules are depicted in Figure 3b. By introducing our integer-to-integer filters for the biorthogonal graph filterbank we obtain a lossy-to-lossless compression scheme using graph wavelets.

It is important to note that this manuscripts aims to study graph wavelet transforms when applied spatially to hyperspectral images. For that reason, we have followed a minimalist approach to designing our overall compression scheme. By avoiding complex quantization or entropy encoding stages, with complex overall interactions, we focus our efforts to the proposed graph transforms and to measure their performance with less interference. While a more complex integration of post-transform encoder parts would be possible, due to the adaptive nature of the biorthogonal graph filterbanks, it is expected that these transforms perform competitively, regardless of the probabilistic characteristics of the pre-transform data. For theoretical result on transform performance, readers can see Reference [25],

where it is proven that the GFT optimally decorrelates images following a Gauss-Markov random field model.

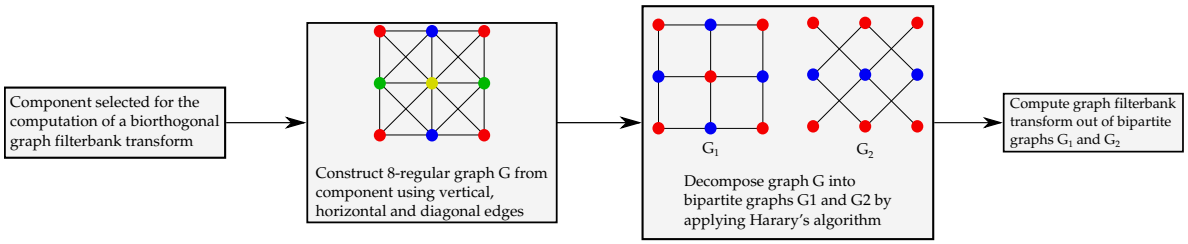
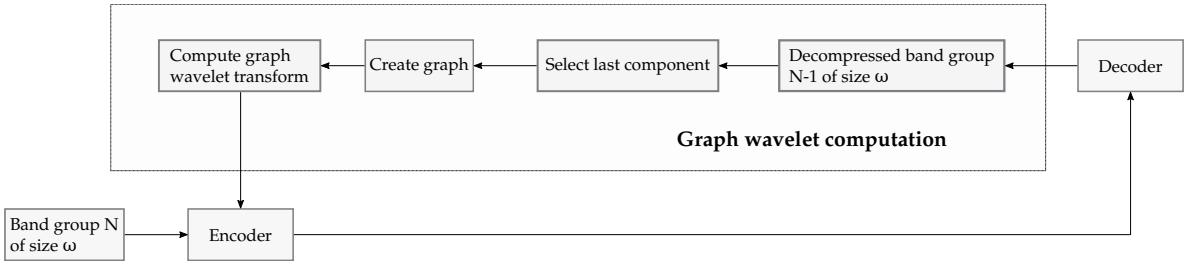
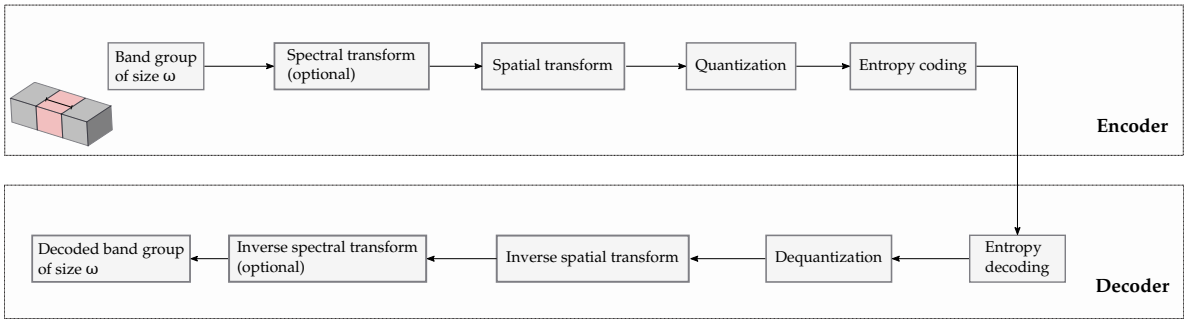


Figure 2. Construction of the bipartite graphs out of a selected component leading to the calculation of a graph filterbank transform.



(**a**) Computation of graph wavelet filterbank transform



(**b**) Compression scheme

Figure 3. Schematics of the compression scheme used.

3.2. TERM factorization of GraphBior

In order for the proposed graph compression scheme, based on the biorthogonal graph filterbanks, to develop a lossless behavior we require an appropriate transform. Specifically, an integer-to-integer spectral graph wavelet transform. One of our solutions is to apply the TERM factorization process and the computational scheme introduced in Reference [35] on the GraphBior transform matrix.

However, one cannot decompose any arbitrary square transform matrix through TERM. The necessary condition in order to apply such a factorization dictates that the determinant of the transform should be strictly equal to 1. By design the GraphBior transform is reversible, which means that the determinant of the GraphBior analysis filter $T_a \in \mathbb{R}^{|V|\times|V|}$ is equal to an arbitrary non zero real number d.

We scale the elements of T_a by $d^{-1/|V|}$, without the loss of graph spectral information, to force its determinant to 1. Therefore we create a new matrix $T'_a = T_a \cdot d^{-1/|V|}$ whose determinant is equal to 1 and can follow the decomposition and computational scheme of Reference [35]. Applying the TERM factorization to the transform T'_a and by following the computation scheme introduced in Reference [35] we manage to create an integer-to-integer variation of the GraphBior transform. We shall abbreviate our proposed TERM GraphBior transform as IGB (integer GraphBior).

This factorization procedure raises the concern for certain drawbacks, specifically if we consider the large size of the GraphBior transform matrix. These square nonsingular matrices have as many rows (or columns) as the total number of pixels in each hyperspectral component. Due to its cubical computational complexity, the factorization of a GraphBior transform for an entire component, through TERM, is computationally intensive. As a first measure we use low order polynomials to compute the GraphBior filters. For that reason we use the GraphBior(1,1) rather than the better performing GraphBior(5,5) transform, since the former produces a sparser transform matrix. Additionally we split each component that is used to create the transform into smaller square tiles in an effort to accelerate the factorization process.

3.3. Tiling

In order to alleviate the time complexity from the TERM factorization, we developed a divide and conquer method. We initially split the component that is used to derive the graph wavelet transform into multiple small square tiles. From each tile, we compute its corresponding GraphBior transform to which we apply the TERM factorization. For the selection of a suitable tile size, in this subsection we study the effect on the performance of our compression scheme with respect to the tile size. Therefore we have experimented with IGB on our compression scheme, with no spectral transforms and using 3 different tile sizes (8 by 8, 16 by 16 and 32 by 32 square tile sizes). We tested the performance of IGB on the hyperspectral images, listed in Table 1, that contain several hyperspectral scenes available at the CCSDS website [36]. In Table 2 we display the entropy, in bits per pixels per component (bpppc), at which IGB becomes lossless, while using $\omega = 2$. We observe that the entropy where the IGB transform becomes lossless decreases as the tile size increases. This first observation encourages us to use a larger tiles for the lossless case. The results from Figure 4 lead to the same conclusions when testing the lossy performance of IGB. Specifically, in Figure 4 we display the performances of IGB when using tiles of 8 by 8 and 16 by 16 pixels, relative to using larger tiles of 32 by 32 pixels. Larger tile sizes are not included in our experiments as we are limited by the high computational complexity of the TERM decomposition. In Figure 5 we have repeated the same experiment on other hyperspectral images. Regarding the two AVIRIS images, due to their much larger spectral size, we have not experimented with the largest tile option as we have done with the Landsat image. The experimental conclusions also agree with our intuitive interpretation of the graph biorthogonal filterbanks since when using larger tiles, the graph can exploit spatial redundancies on a bigger area of each component. This is because the graph edges that connect adjacent nodes that belong to different tiles are not utilized. Since the number of discarded edges rises as the tiles becomes smaller so does the performance of IGB deteriorate. Therefore it is beneficial to keep the sizes of the tiles as large as possible but small enough to allow a fast TERM factorization. Luckily this solution can be parallelized since we could process every tile of every component of the same group at the same time.

Table 1. The hyperspectral images with their dimensions used in our experiments.

Name	Instrument	Calibrated	Across-Track	Along-Track	Spectral Dimension
Yellowstone sc. 0 cal.	AVIRIS	Yes	512	512	224
Yellowstone sc. 0 raw	AVIRIS	No	512	512	224
Lake Monona	Hyperion	Yes	512	256	242
Mt. St. Helens	Hyperion	Yes	512	256	242
Agriculture	Landsat	No	512	512	6

Table 2. Rates at which integer GraphBior (IGB) achieves a lossless compression using $\omega = 2$ for various tile sizes and multiple images. Units are in bpppc.

Image	IGB		
	Tiles of 8 by 8	**Tiles of 16 by 16**	**Tiles of 32 by 32**
Yellowstone sc. 0 cal.	7.14	6.95	-
Yellowstone sc. 0 raw	9.15	9.02	-
Lake Monona	6.40	6.31	6.26
Mt. St. Helens	6.78	6.68	6.63
Agriculture	4.39	4.27	4.21

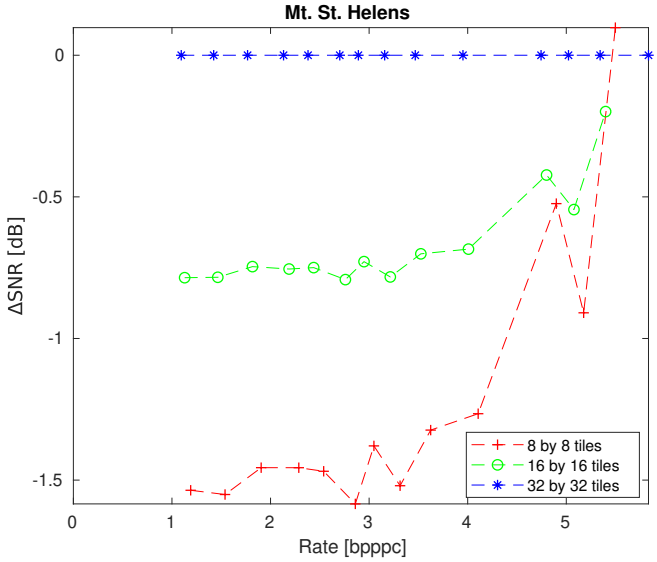


Figure 4. Relative rate-distortion plots for IGB, using $\omega = 2$, when varying the tile size. Results are relative to those of the largest tile size.

3.4. Integer-to-Integer Spectral Graph Lifting

The main disadvantage of the IGB arises from the computational complexity of the TERM factorization step. As we have seen in the previous section although the tiling strategy reduces the time complexity for the TERM factorization, the performance of our compression scheme decreases in relation to the size of the tiles. For that reason we introduce a second integer-to-integer graph filterbank transform using the spectral graph lifting filters [34].

These filters are arranged into Type 1 and 2 polyphase transform matrices (PTM) which have an upper and lower triangular structure respectively. The final transform matrix is calculated by multiplying alternatively Type 1 PTMs with Type 2 PTMs. One can notice that both types of PTMs are unit triangular rectangular matrices, which means that by following the TERM filtering process we can modify the spectral graph lifting filters into integer-to-integer transforms. Due to their particular structure, by introducing rounding operations after applying each PTM matrix multiplication, the transform becomes integer and reversible. We experiment with the integer-to-integer spectral graph lifting using the quad kernel (ISGL_Q) and the integer-to-integer spectral graph lifting using the dual kernel (ISGL_D) designs [41].

It should be noted that the computational complexity of this transform is comparable to the one of the GraphBior transform and can be easily computed from an entire component without the need of tiling operations. Specifically the computational complexity of the costlier ISGL_Q is comparable to the one of GraphBior(5,5) since the former requires a computation of a 10th and 13th degree matrix polynomial, whereas the latter one of 10th and one of 11th. This means that by implementing the filtering process with Chebyshev polynomials [28] one can use the same order M to approximate

similarly well the GraphBior(5,5) as well as the $ISGL_Q$ filters resulting in a computational complexity of $O(M|E|)$ per filter, where E is the set of edges of the graph. By performing the appropriate modifications, our proposed $ISGL_Q$ compresses and decompresses the calibrated AVIRIS image from Table 1 in 4.5 min, whereas the IGB required more than 1 h to successfully perform the same action on a computer using an Intel Core i7-7700HQ CPU and 16GB of RAM.

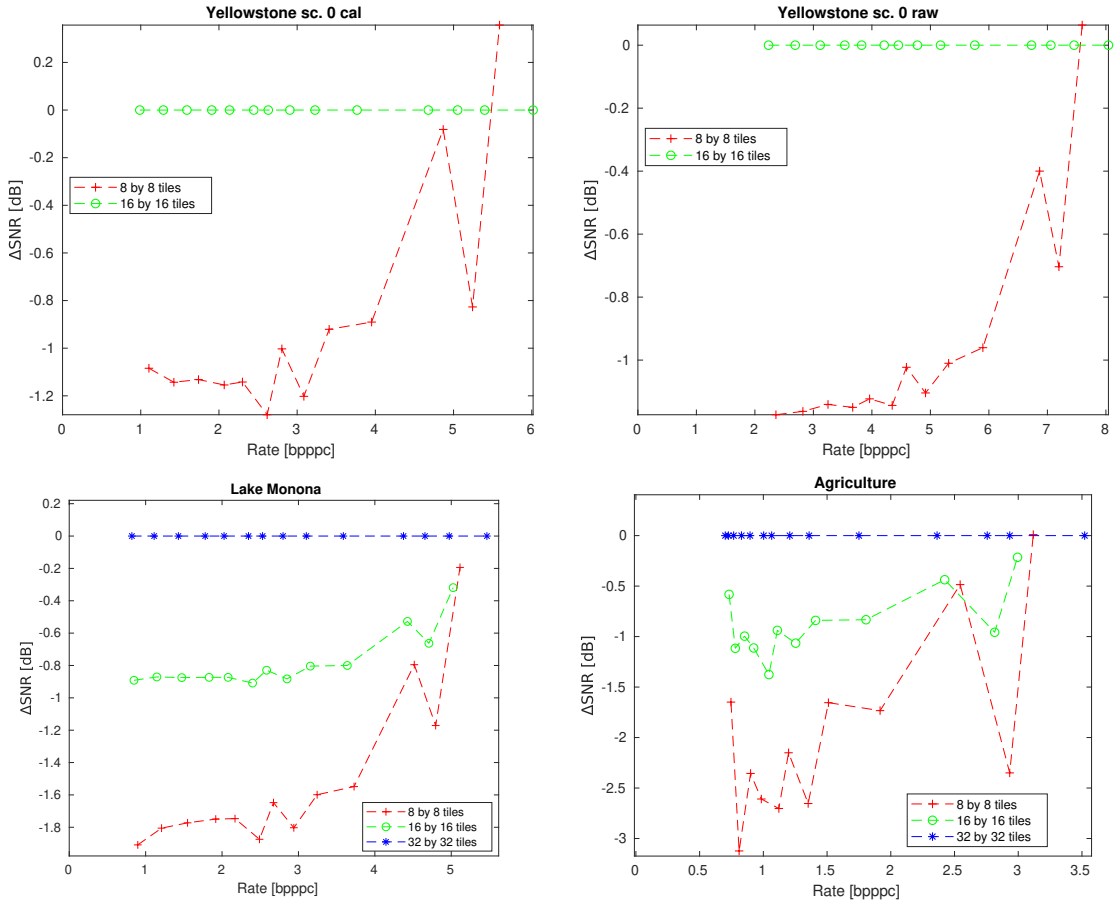


Figure 5. Relative rate-distortion plots for IGB, using $\omega = 2$, when varying the tile size, for multiple images. Results are relative to those of the largest tile.

4. Experimental Results

In this section we describe the setting of our experiments and analyze our results. We provide several comparisons testing the performance of our compression scheme using our proposed transforms, in both progressive lossy-to-lossless and lossless settings. We should note that regarding the rate computations, we compute the average of the entropies of each compressed component instead of using any particular entropy encoder to discard any bias introduced by an entropy encoder fine-tuned for a specific transform. All experiments have been performed using a prototype encoder implementation in Matlab R2017b. Initially we experiment solely on spatial transforms. We compare our proposed transforms against the GraphBior transform, with and without dividing the components into tiles as well as the reversible 5/3 integer wavelet transform (DWT). All GraphBior transforms are using the maximally flat GraphBior(1,1) filters. The spectral graph lifting designs are using the maximum flatness dual (with the $B_{7,3}$ Parametric-Bernstein-Polynomial for $ISGL_D$) and quad (with the $B_{3,1}$ Parametric-Bernstein-Polynomial for $ISGL_Q$) kernel designs without normalization parameters [41]. We provide further comparisons between the spatial transforms by including spectral transforms such as the Reversible Karhunen Loeve Transform (RKLT) [42] and the DWT.

We have repeated our experiments on crops of the Aviris Yellowstone scene 00, the Hyperion images Lake Monona and Mt. St. Helens and the Landsat Agriculture image. The dimensions of the cropped images that have been used are displayed in Table 1. More precisely, we have discarded the last rows and columns of each component for both Aviris as well as the Landsat scenes. Regarding the Hyperion images we have discarded the last columns and the first 1350 and 760 rows for Lake Monona and Mt. St. Helens respectively.

Next we explore the version of the compression scheme where we include a spectral transform on each band group before applying the spatial transform. The graph transforms are always computed from the luminance values of components and not the spectrally transformed results. We experiment with the Reversible Karhunen Loeve Transform (RKLT) [42] and DWT as spectral transforms. The spatial transforms compared in this experiment are the proposed IGB, the tiled GraphBior, the DWT, the ISGL_Q and ISGL_D. In this final experiment, we also include a comparison of our overall compression scheme with the CCSDS-123.0-B-2 standard, tuned according to Reference [19].

4.1. Parameter Variations

We first study the effect that different values of ω have on IGB, without taking into account spectral transforms. To search for the most beneficial choices of ω to IGB we use tiles of large size.

In Figure 6 the relative rate-distortion plot displays the decrease of performance when larger values of ω are compared against the smallest $\omega = 2$. One can clearly conclude that it is beneficial to choose smaller values of the parameter ω when no spectral transform is used. We repeat the experiment with several other hyperspectral images and display the results in Figure 7. Specifically, it is for $\omega = 2$ that tends to lead to optimal results in all cases. This result agrees with our expectations since components closer in the spectral dimension tend to have high correlation. Thus, the component that has been used to create the IGB transform will capture sometimes more accurately the spatial redundancies of its neighboring component rather than of one located further away in the spectral dimension. Though this can lead to detrimental results for low ω values, when high quantization occurs. As we can see from Figures 6 and 7c,d at low rates, larger parameters ω outperform lower ones.

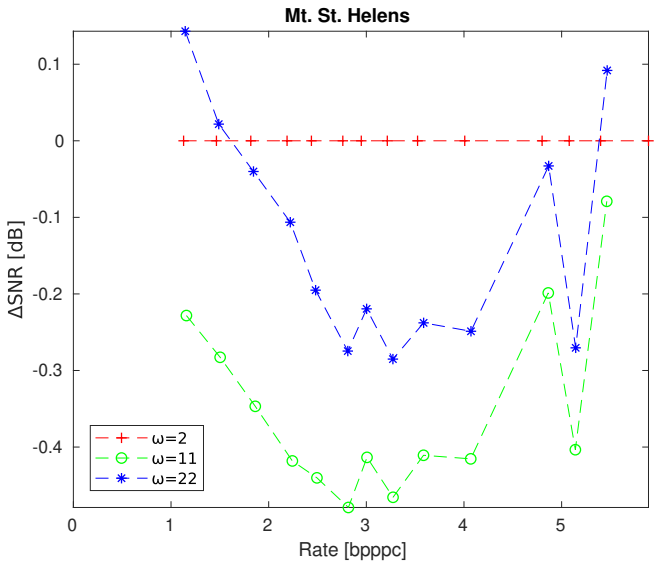


Figure 6. Relative rate-distortion plot for IGB using tiles of 16 by 16 when varying ω. The reference curve coincides with tuning the parameter $\omega = 2$.

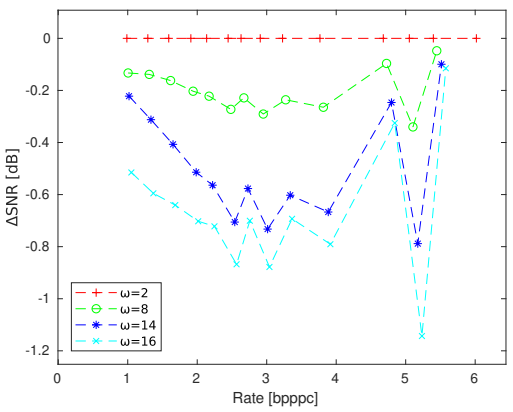
This could be explained by high quantization, when the surviving features preserved in a decompressed component are general enough to construct a graph that represents sufficiently well a larger number of components. Furthermore, the smaller the parameter ω, the higher the probability of outlier components (ones that do not share high correlation with neighboring components such

as dead bands) to be used for the graph creation, which may result in poor reconstruction results of components of an entire band group.

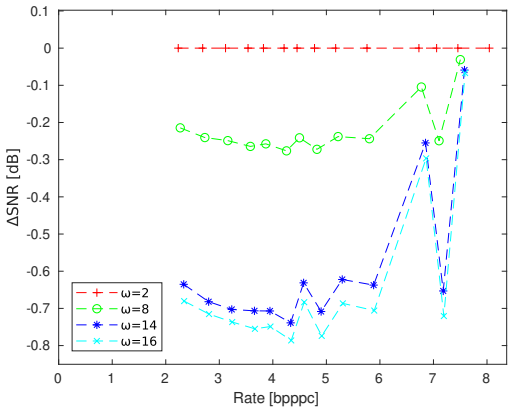
From Table 3 we also observe that the entropy where the compression scheme using the IGB transform becomes lossless, increases with ω. This explains why at high rates, higher ω values can be seen to sometimes perform better. This is also noticeable in classical reversible integer wavelet transforms. Their rate distortion curves tend to plateau close to the bitrate at which they achieve a reversible compression. As a result, sometimes, they are surpassed by the distortion curves corresponding to methods that become reversible at higher bitrates (since they still continue to grow with a higher gradient).

Table 3. Rates at which IGB achieves a lossless compression for different values of ω. The tiles size is set to 16 by 16. The spectral size of the hyperspectral image should be a multiple of ω, resulting in empty cells. Units are in bpppc.

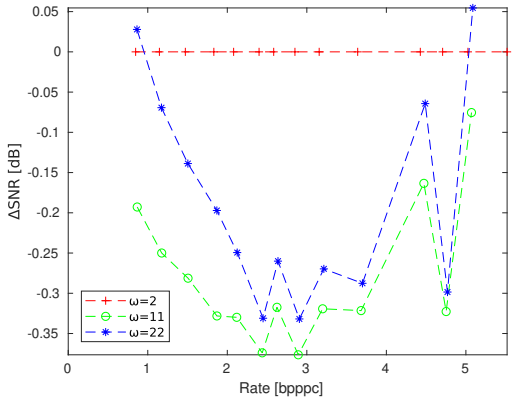
Image	**IGB**						
	$\omega = 2$	$\omega = 3$	$\omega = 8$	$\omega = 11$	$\omega = 14$	$\omega = 16$	$\omega = 22$
Yellowstone sc. 0 cal.	6.95		7.00		7.06	7.14	
Yellowstone sc. 0 raw	9.02		9.06		9.14	9.15	
Lake Monona	6.31			6.36			6.37
Mt. St. Helens	6.68			6.75			6.75
Agriculture	4.27	4.39					



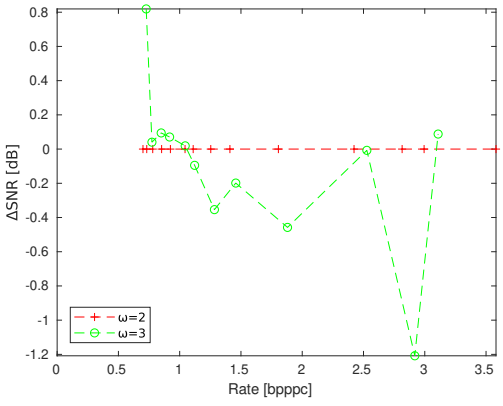
(**a**) Yellowstone sc. 0 cal



(**b**) Yellowstone sc. 0 raw



(**c**) Lake Monona



(**d**) Agriculture

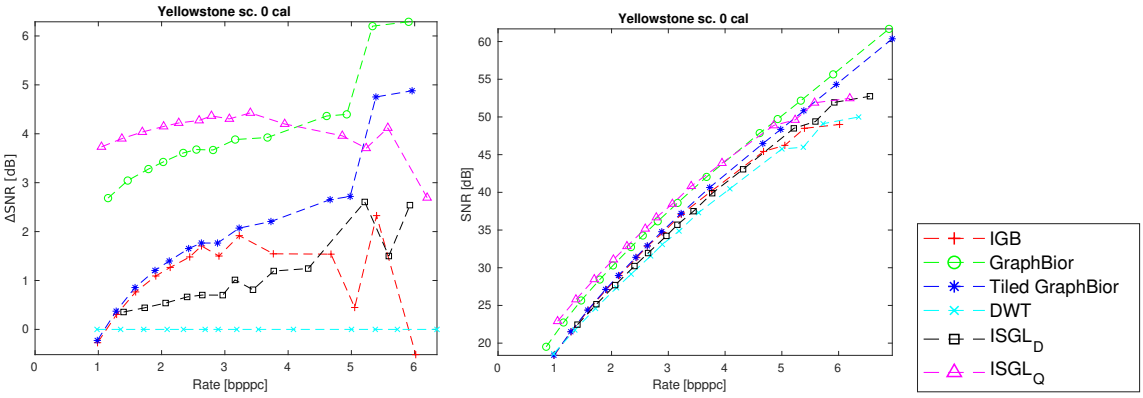
Figure 7. Relative rate-distortion plots for IGB using tiles of 16 by 16 when varying the parameter ω for multiple images. Results are relative to those of smallest value of ω.

4.2. Spatial Transformations

Adopting our previous findings regarding the parameter ω as well as the size of the tiles, we compare our proposed reversible transforms against the DWT and the GraphBior with and without using tiles. All transforms are applied spatially whereas spectral transforms are not included in the subsequent experiments. It should be noted that all spatial transforms are tested using the compression scheme introduced in Section 3.

For this experiment, the rate-distortion plot, as well as the relative rate-distortion plot using the DWT transform as reference can be observed in Figure 8a,b respectively. From the rate-distortion plots of Figure 8 we observe that the ISGL_Q performs the best for low and medium rates, whereas GraphBior provides the best results for high rates. It is important to mention, though, that in most cases IGB slightly outperforms the DWT as well as the ISGL_D at low rates. Moreover, although in general, tiled GraphBior outperforms IGB, at low rates, both perform similarly. This same behavior is also observed between classical discrete wavelet transforms and their reversible counterparts. Given the previous observation and since the results of IGB improve as the tile size increases, one could speculate that a IGB that does not use any tiles would perform similarly to GraphBior for low rates. This experiment has been repeated on different hyperspectral scenes and the results can be observed in Figure 9.

In Table 4 we can see the rates at which the IGB, DWT, ISGL_Q and ISGL_D achieve a lossless compression. We also observe that in most cases the entropy where IGB achieves a lossless compression is the lowest.



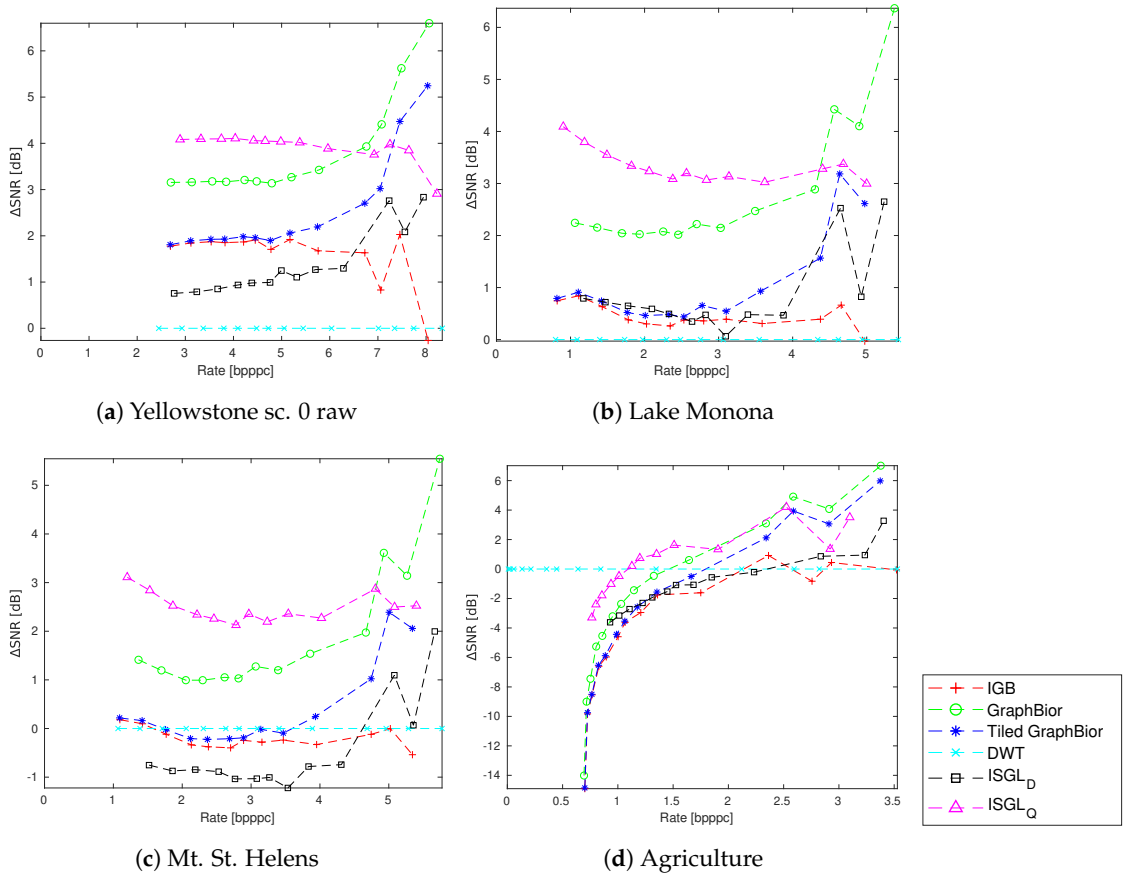
(**a**) Relative rate-distortion plot with respect to DWT.

(**b**) Rate-distortion plot.

Figure 8. Comparison of spatial transforms. Tiles of 16 by 16 and $\omega = 2$ were used.

Table 4. Rates at which each integer transform achieves a lossless compression. No spectral transform is used. The parameter ω is set to 2. The IGB transform uses tiles of 16 by 16 for the Yellowstone images and 32 by 32 for Lake Monona, Mt. St. Helens and Agriculture. Units are in bpppc.

Image	Transform			
	IGB	**DWT**	**ISGL_D**	**ISGL_Q**
Yellowstone sc. 0 cal.	6.95	7.29	7.44	7.11
Yellowstone sc. 0 raw	9.02	9.32	9.47	9.19
Lake Monona	6.26	6.23	6.50	6.28
Mt. St. Helens	6.63	6.57	6.93	6.67
Agriculture	4.21	4.37	4.67	4.37



(**a**) Yellowstone sc. 0 raw (**b**) Lake Monona

(**c**) Mt. St. Helens (**d**) Agriculture

Figure 9. Relative rate-distortion plots comparing spatial transforms using $\omega = 2$ for multiple images. Results are relative to DWT. The tiles sizes are 16 by 16 for the Yellowstone sc. 0 raw image and 32 by 32 for the rest.

4.3. Spectral and Spatial Transformations

Our comparisons on strictly spatial transforms are ensued by experiments that include the DWT and the RKLT as spectral transforms, in the compression scheme mentioned in Section 3. Succeeding the spectral transform, we then compare our proposed reversible transforms against the DWT and the tiled GraphBior applied in the spatial dimension of the hyperspectral images. Our overall compression scheme including classical spectral and graph spatial transforms is also compared against the CCSDS-123.0-B-2 standard.

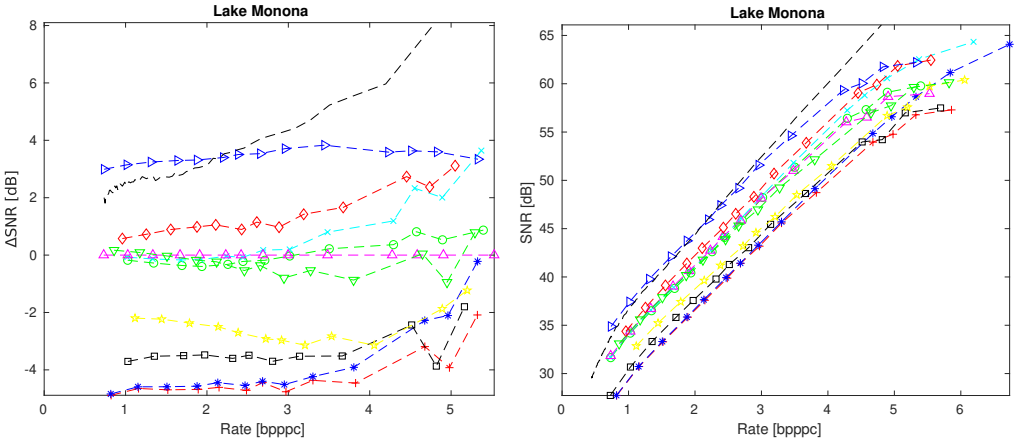
In Figure 10a,b we present the rate-distortion plot as well as the relative rate-distortion plot. The results are relative to applying a spectral RKLT along the entire hyperspectral image, followed by a spatial DWT.

Regarding the comparison between the transforms used in our compression scheme, we observe that a spectral RKLT followed by the proposed ISGL_Q or ISGL_D transforms perform the best. Several additional comparisons are done on multiple hyperspectral images and from our results in Figure 11 we observe that the spectral RKLT and the spatial ISGL_Q systematically outperforms the spectral RKLT and spatial DWT, mostly at medium to high rates. The spectral RKLT and spatial ISGL_D, also, usually surpasses the reference method but provides less important results when compared to using the spatial ISGL_Q instead. On the other hand the spectral RKLT and the spatial IGB is almost always providing worse results when compared to the spectral RKLT and the spatial DWT.

Our conclusions from the comparison of the spatial transforms done in Section 4.2 are also evident in our current experiment. Regardless of the spectral transform, the spatial IGB coincides with the spatial tiled GraphBior for low entropy values. When we apply DWT as a spectral transform, out of the spatial transforms, the ISGL_Q performs the best followed mostly by the DWT and the ISGL_D.

From Table 5, we observe that mostly the spectral RKLT and the spatial DWT achieve a lossless compression at lower rates. Only in the case of the Hyperion images (Lake Monona and Mt. St. Helens) the spectral RKLT followed by the spatial $ISGL_Q$ performs better.

When comparing the compression efficiency of the latest CCSDS-123.0-B-2 standard against our overall proposed compression scheme, we observe the usual pattern where transform-based methods tend to perform better at lower rates while the CCSDS standard yields better results at mid to high rates. In Figures 10 and 11, a spectral RKLT followed by a spatial DWT often outperforms CCSDS for low rates. Furthermore, CCSDS results are also improved by the proposed compression scheme using a spectral RKLT and a spatial $ISGL_Q$ for low-rate results in Figure 11a,b. In the mid to high rate regions, the CCSDS standard clearly outperforms all transform-based methods. In addition, for pure lossless compression, the CCSDS standard consistently achieves best results (Table 5). It should be noted though that the CCSDS standard is a highly refined method, whereas the proposed compression scheme here presented is mainly designed to compare the graph transforms. Thus, it lacks many of the techniques that the CCSDS standard incorporates, such as quantization enhancements, noise tolerance or entropy encoding with adaptive statistics.



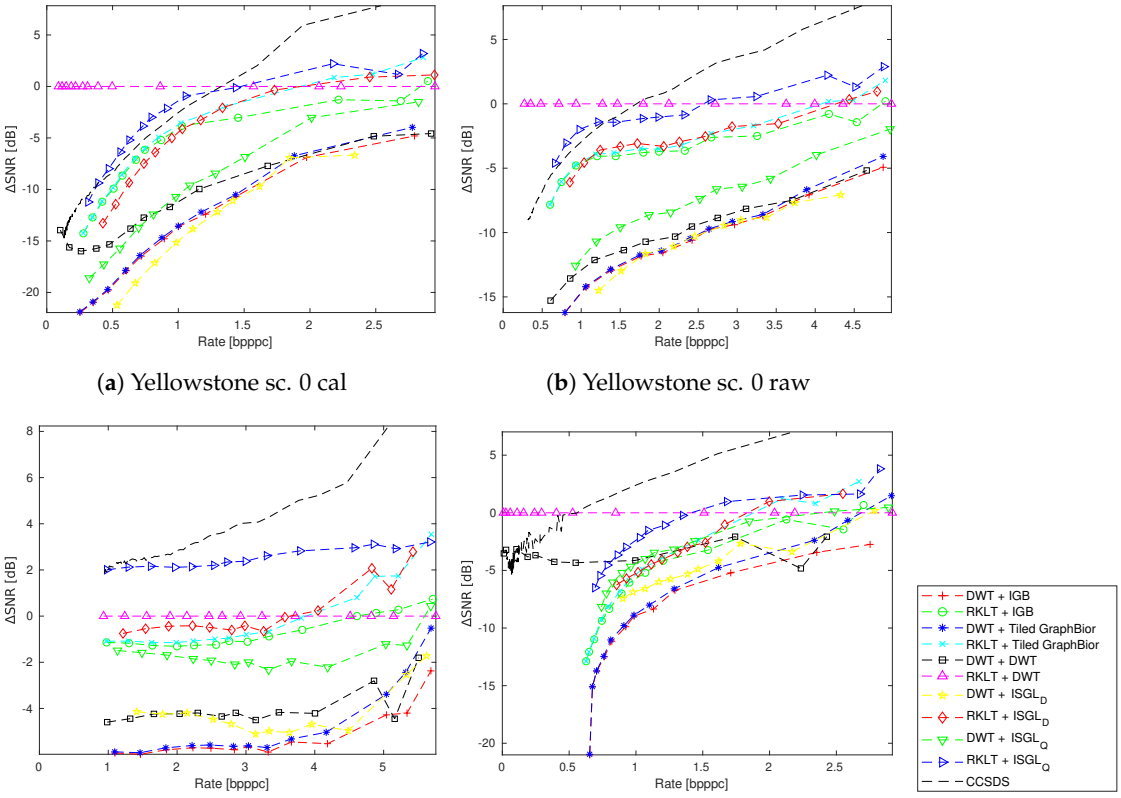
(**a**) Relative rate-distortion plot with respect to the spectral RKLT followed by the spatial DWT.

(**b**) Rate-distortion plot

Figure 10. Comparison of spectral + spatial transforms. Tiles of 16 by 16 and $\omega = 11$ was used.

Table 5. Rates at which each of the integer transforms achieves a lossless compression. RKLT and DWT are used as spectral transforms. For the methods that use tiles, their size has been set to 16 by 16. The Yellowstone images are evaluated at $\omega = 8$, the Lake Monona and Mt. St. Helens are evaluated at $\omega = 11$ and the Agriculture image is evaluated at $\omega = 3$. For each column, the spectral transform is mentioned first followed by the spatial transform (spectral transform + spatial transform). Units are in bpppc.

Image	Transforms								CCSDS
	DWT + IGB	RKLT + IGB	DWT + DWT	DWT + ISGL$_D$	DWT + ISGL$_Q$	RKLT + DWT	RKLT + ISGL$_D$	RKLT + ISGL$_Q$	
Yellowstone sc. 0 cal.	5.03	4.41	4.73	5.36	5.05	3.74	4.65	4.36	4.04
Yellowstone sc. 0 raw	7.17	6.47	6.97	7.54	7.25	5.93	6.72	6.44	6.19
Lake Monona	6.69	6.19	6.56	6.89	6.66	6.35	6.34	6.13	6.10
Mt. St. Helens	7.06	6.52	6.90	7.32	7.05	6.58	6.71	6.47	6.37
Agriculture	4.21	3.96	3.96	4.63	4.34	3.68	4.37	4.08	3.62



(**a**) Yellowstone sc. 0 cal

(**b**) Yellowstone sc. 0 raw

(**c**) Mt. St. Helens

(**d**) Agriculture

Figure 11. Relative rate-distortion plots comparing spectral + spatial transforms for multiple images. Results are relative to the spectral RKLT followed by the spatial DWT. Tiles of 16 by 16 were used. The parameter ω is set to 8 for Yellowstone sc. 0 cal., to 11 for Mt. St. Helens and to 3 for Agriculture.

5. Conclusions

In this paper we study the suitability of graph transforms for lossy-to-lossless hyperspectral compression. The adopted compression scheme organizes the components into packets, called band groups and transforms each one of them through graph wavelet filterbanks. We introduce two spatial, integer-to-integer, biorthogonal graph filterbank transforms. The first transform is calculated by applying a TERM factorization on the GraphBior filterbank, whereas the second one is designed by modifying the spectral graph lifting transform [34]. The high computational complexity of the TERM decomposition is addressed and is solved through processing the GraphBior transform in tiles. Our theoretical interpretations as well as our experimental results show that it is advantageous to use larger tiles and give us valuable insight about the choice of sizes of band groups. Our experiments without spectral transformations suggest that it is preferable to use small band groups, while our

integer-to-integer transforms outperform the DWT in the lossy regime. Further-more we show that one of our integer transforms performs similarly to the DWT for the case of lossless compression. Additional experimental results including spectral transforms on each bandgroup show that our proposition improves on the results obtained by using the spectral RKLT along all the spectral dimension of the hyperspectral image followed by the spatial DWT in the lossy setting, as well as, in some cases, at the lossless one.

Author Contributions: Conceptualization, D.E.O.T., K.C., I.B. and J.S.-S.; methodology, D.E.O.T., K.C., I.B. and J.S.-S.; software, D.E.O.T.; validation, D.E.O.T., I.B. and J.S.-S.; formal analysis, D.E.O.T., K.C., I.B. and J.S.-S. ; investigation, D.E.O.T., K.C., I.B. and J.S.-S.; resources, D.E.O.T., K.C., I.B. and J.S.-S.; data curation, D.E.O.T., I.B. and J.S.-S.; writing–original draft preparation, D.E.O.T.; writing–review and editing, D.E.O.T., I.B. and J.S.-S.; visualization, D.E.O.T., I.B. and J.S.-S.; supervision, I.B. and J.S.-S.; project administration, I.B. and J.S.-S.; funding acquisition, I.B. and J.S.-S.

Funding: This research was funded by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund under grants RTI2018-095287-B-I00 and TIN2015-71126-R (MINECO/FEDER, UE) and BES-2016-078369 (Programa Formación de Personal Investigador), and by the Catalan Government under grant 2017SGR-463.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liang, H.; Li, Q. Hyperspectral imagery classification using sparse representations of convolutional neural network features. *Remote Sens.* **2016**, *8*, 99. [CrossRef]
2. Thompson, D.R.; Green, R.O.; Keymeulen, D.; Lundeen, S.K.; Mouradi, Y.; Nunes, D.C.; Castaño, R.; Chien, S.A. Rapid spectral cloud screening onboard aircraft and spacecraft. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 6779–6792. [CrossRef]
3. Zhang, X.; Liu, F.; He, Y.; Li, X. Application of hyperspectral imaging and chemometric calibrations for variety discrimination of maize seeds. *Sensors* **2012**, *12*, 17234–17246. [CrossRef] [PubMed]
4. Wang, X.; Niu, R. Spatial forecast of landslides in three gorges based on spatial data mining. *Sensors* **2009**, *9*, 2035–2061. [CrossRef]
5. Tang, X.; Pearlman, W.A. Three-dimensional wavelet-based compression of hyperspectral images. In *Hyperspectral Data Compression*; Springer: Boston, MA, USA, 2006; pp. 273–308.
6. Fowler, J.E.; Rucker, J.T. Three-dimensional wavelet-based compression of hyperspectral imagery. In *Hyperspectral Data Exploitation: Theory and Applications*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007; pp. 379–407.
7. Karami, A.; Yazdi, M.; Mercier, G. Compression of hyperspectral images using discerete wavelet transform and tucker decomposition. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2012**, *5*, 444–450. [CrossRef]
8. Tang, X.; Pearlman, W.A.; Modestino, J.W. Hyperspectral image compression using three-dimensional wavelet coding. In *Image and Video Communications and Processing 2003*; International Society for Optics and Photonics: Bellingham, WA, USA, 2003; Volume 5022, pp. 1037–1047.
9. Du, Q.; Fowler, J.E. Hyperspectral image compression using JPEG2000 and principal component analysis. *IEEE Geosci. Remote Sens. Lett.* **2007**, *4*, 201–205. [CrossRef]
10. Penna, B.; Tillo, T.; Magli, E.; Olmo, G. Transform coding techniques for lossy hyperspectral data compression. *IEEE Geosci. Remote Sens. Lett.* **2007**, *45*, 1408–1421. [CrossRef]
11. Penna, B.; Tillo, T.; Magli, E.; Olmo, G. A new low complexity KLT for lossy hyperspectral data compression. In Proceedings of the 2006 IEEE International Symposium on Geoscience and Remote Sensing, Denver, CO, USA, 31 July–4 August 2006; pp. 3525–3528.
12. Galli, L.; Salzo, S. Lossless hyperspectral compression using KLT. In Proceedings of the 2004 IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2004), Anchorage, AK, USA, 20–24 September 2004; Volume 1.
13. Rizzo, F.; Carpentieri, B.; Motta, G.; Storer, J.A. Low-complexity lossless compression of hyperspectral imagery via linear prediction. *IEEE Signal Process. Lett.* **2005**, *12*, 138–141. [CrossRef]
14. Pizzolante, R.; Carpentieri, B. Multiband and lossless compression of hyperspectral images. *Algorithms* **2016**, *9*, 16. [CrossRef]

15. Abrardo, A.; Barni, M.; Magli, E. Low-complexity predictive lossy compression of hyperspectral and ultraspectral images. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; pp. 797–800.
16. Klimesh, M. Low-complexity adaptive lossless compression of hyperspectral imagery. In Proceedings of the Satellite Data Compression, Communications, and Archiving II, San Diego, CA, USA, 13–17 August 2006; Volume 6300, p. 63000N.
17. Klimesh, M.; Kiely, A.; Yeh, P. Fast lossless compression of multispectral and hyperspectral imagery. In Proceedings of the 2nd International Workshop on On-Board Payload Data Compression (OBPDC), Toulouse, France, 28–29 October 2010; Volume 8, pp. 28–29.
18. Consultative Committee for Space Data Systems. *Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression. Recommendation for Space Data System Standards, CCSDS 123.0-B-2*; CCSDS: Washington, DC, USA, 2019.
19. Blanes, I.; Kiely, A.; Hernández-Cabronero, M.; Serra-Sagristà, J. Performance Impact of Parameter Tuning on the CCSDS-123.0-B-2 Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression Standard. *Remote Sens.* **2019**, *11*, 1390. [CrossRef]
20. Shuman, D.I.; Narang, S.K.; Frossard, P.; Ortega, A.; Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.* **2013**, *30*, 83–98. [CrossRef]
21. Ortega, A.; Frossard, P.; Kovačević, J.; Moura, J.M.; Vandergheynst, P. Graph signal processing: Overview, challenges, and applications. *Proc. IEEE* **2018**, *106*, 808–828. [CrossRef]
22. Zeng, J.; Cheung, G.; Chao, Y.H.; Blanes, I.; Serra-Sagristà, J.; Ortega, A. Hyperspectral image coding using graph wavelets. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017.
23. Cheung, G.; Magli, E.; Tanaka, Y.; Ng, M.K. Graph spectral image processing. *Proc. IEEE* **2018**, *106*, 907–930. [CrossRef]
24. Hu, W.; Cheung, G.; Ortega, A.; Au, O.C. Multiresolution graph fourier transform for compression of piecewise smooth images. *IEEE Trans. Image Process.* **2014**, *24*, 419–433. [CrossRef] [PubMed]
25. Zhang, C.; Florêncio, D. Analyzing the optimality of predictive transform coding using graph-based models. *IEEE Signal Process. Lett.* **2012**, *20*, 106–109. [CrossRef]
26. Narang, S.K.; Ortega, A. Lifting based wavelet transforms on graphs. In Proceedings of the APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, Sapporo, Japan, 4–7 October 2009; pp. 441–444.
27. Coifman, R.R.; Maggioni, M. Diffusion wavelets. *Appl. Comput. Harmonic Anal.* **2006**, *21*, 53–94. [CrossRef]
28. Hammond, D.K.; Vandergheynst, P.; Gribonval, R. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmonic Anal.* **2011**, *30*, 129–150. [CrossRef]
29. Narang, S.K.; Ortega, A. Perfect reconstruction two-channel wavelet filter banks for graph structured data. *IEEE Trans. Signal Process.* **2012**, *60*, 2786–2799. [CrossRef]
30. Narang, S.K.; Ortega, A. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE Trans. Signal Process.* **2013**, *61*, 4673–4685. [CrossRef]
31. Sakiyama, A.; Watanabe, K.; Tanaka, Y.; Ortega, A. Two-Channel Critically Sampled Graph Filter Banks with Spectral Domain Sampling. *IEEE Trans. Signal Process.* **2019**, *67*, 1447–1460. [CrossRef]
32. Li, S.; Jin, Y.; Shuman, D.I. Scalable *M*-Channel Critically Sampled Filter Banks for Graph Signals. *IEEE Trans. Signal Process.* **2019**, *67*, 3954–3969. [CrossRef]
33. Tay, D.B.; Ortega, A. Bipartite graph filter banks: Polyphase analysis and generalization. *IEEE Trans. Signal Process.* **2017**, *65*, 4833–4846. [CrossRef]
34. Tay, D.B.; Ortega, A.; Anis, A. Cascade and Lifting Structures in the Spectral Domain for Bipartite Graph Filter Banks. In Proceedings of the 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Honolulu, HI, USA, 12–15 November 2018; pp. 1141–1147.
35. Hao, P.; Shi, Q. Matrix factorizations for reversible integer mapping. *IEEE Trans. Signal Process.* **2001**, *49*, 2314–2324.
36. Consultative Committee for Space Data Systems, Test Data. Available online: http://cwe.ccsds.org/sls/docs/sls-dc/123.0-B-Info/TestData (accessed on 3 March 2019).
37. Harary, F.; Hsu, D.; Miller, Z. The biparticity of a graph. *J. Graph Theory* **1977**, *1*, 131–133. [CrossRef]

38. Narang, S.K.; Ortega, A. Multi-dimensional separable critically sampled wavelet filterbanks on arbitrary graphs. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 3501–3504.
39. Nguyen, H.Q.; Do, M.N. Downsampling of signals on graphs via maximum spanning trees. *IEEE Trans Signal Process.* **2014**, *63*, 182–191. [CrossRef]
40. Zeng, J.; Cheung, G.; Ortega, A. Bipartite approximation for graph wavelet signal decomposition. *IEEE Trans Signal Process.* **2017**, *65*, 5466–5480. [CrossRef]
41. Tay, D.B.; Zhang, J. Techniques for constructing biorthogonal bipartite graph filter banks. *IEEE Trans Signal Process.* **2015**, *63*, 5772–5783. [CrossRef]
42. Hao, P.; Shi, Q. Reversible integer KLT for progressive-to-lossless compression of multiple component images. In Proceedings of the 2003 International Conference on Image Processing (Cat. No. 03CH37429), Barcelona, Spain, 14–17 September 2003; Volume 1, pp. 1–633.

Chapter 3

Orthogonality and Zero DC Tradeoffs in Biorthogonal Graph Filterbanks

CORE quality factor: B

ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413392

ORTHOGONALITY AND ZERO DC TRADEOFFS IN BIORTHOGONAL GRAPH FILTERBANKS

Dion E.O. Tzamarias[⋆], *Eduardo Pavez*[†], *Benjamin Girault*[†#]
Antonio Ortega[†], Ian Blanes[⋆], Joan Serra-Sagristà[⋆]

[⋆]Universitat Autònoma de Barcelona, Cerdanyola del Vallès, Barcelona, SPAIN
[†] University of Southern California, Los Angeles, California, USA
[#]Université de Rennes, ENSAI, CNRS, CREST-UMR 9194, Rennes, FRANCE

ABSTRACT

Biorthogonal graph wavelet filterbanks, also known as GraphBior, are one of the most popular graph transforms used in image compression, but up to now, they could be designed based on two known admissible fundamental matrices: i) the random walk Laplacian, which heavily penalizes low degree pixels, and ii) the normalized Laplacian, which lacks a zero-DC response. By exploiting a new extension of the admissibility condition in GraphBior we propose a new fundamental matrix with the goal of distributing the errors of GraphBior more uniformly across pixels with different node degrees. Furthermore the proposed matrix preserves high energy compaction linked to the zero-DC GraphBior variation.

Index Terms— graph signal processing, biorthogonal filterbanks, image compression, graph Fourier transform, irregularity-aware graph Fourier transforms

1. INTRODUCTION

Graph structures can be used to represent data on irregular domains [1, 2] while graph signal processing extends concepts from classical signal processing to the graph domain. As an example, in analogy to classical wavelet filterbanks, a family of graph filterbanks have been proposed and applied to design image compression transforms [1, 3, 4, 5, 6, 7]. One such transform is the biorthogonal graph filterbank (GraphBior) [8]. GraphBior was proposed as an extension of the orthogonal graph filterbanks [9] by improving their spectral localization through the computation of polynomial filters of the normalized Laplacian matrix. However, a drawback of the normalized Laplacian is its non zero-DC property [8], i.e., its lowest frequency signal is not a constant signal, which may affect its performance negatively in the context of image compression. In order to rectify this problem, a zero-DC GraphBior transform was proposed, where each signal entry is scaled by a different factor (dependent on node degree) before applying the analysis filters [8]. This pre-processing is reversed after the signal is reconstructed with the synthesis filterbank. This procedure has been shown to be equivalent to defining GraphBior filters as polynomials of the random walk Laplacian [10].

By replacing the normalized Laplacian with the random walk Laplacian in GraphBior, a significant gain in compression is observed, mainly due to superior energy compaction. Although there is a clear benefit in using the zero-DC GraphBior design, the random walk Laplacian also introduces some drawbacks. For example, compared to the normalized Lapliacian, as noted in [8] and illustrated by Figs. 1a and 1b, the random walk Laplacian filterbank produces accumulation of large errors in high gradient areas of images in an image compression application. Although the overall reconstructed image distortion is equal in both cases, one can clearly see that the details on the edges and faces of the coins are less visible in the case of the zero-DC GraphBior while the smooth background is more accurate. To tackle this problem, an edge aware design was proposed such that filtering would be restricted across boundaries [11]. This procedure alleviates this particular drawback of the zero-DC GraphBior, but does not identify or solve the root cause.

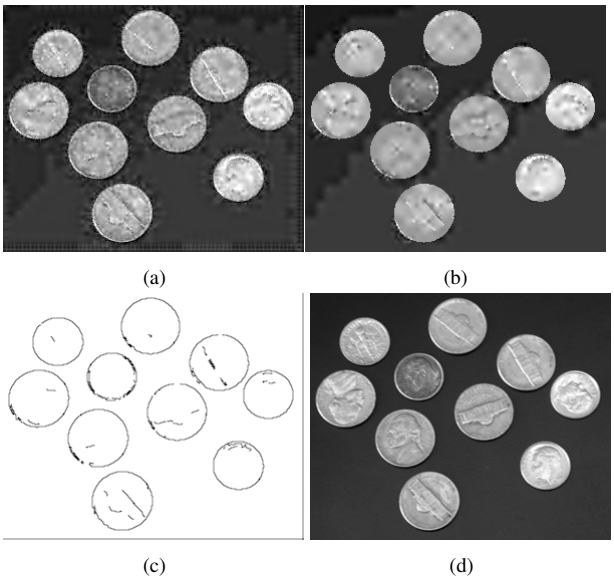


Fig. 1: Reconstruction of Coins at 27.27dB using: (a) Normalized, (b) Random Walk Laplacian, (c) visualization of degrees of Coins, and (d) original coin image. The graph is constructed so that weights are smaller in high gradient regions to avoid smoothing image edges.

The eigenvectors of the random walk Laplacian are orthogonal with respect to the *degree inner product*, defined for any two vectors $\mathbf{x}$ and $\mathbf{y}$ as $\mathbf{y}^\top \mathbf{D}\mathbf{x}$, where $\mathbf{D}$ is the degree matrix [12]. Recently, [10] shows that the zero DC GraphBior filterbanks are also approximately orthogonal in the degree inner product and not just biorthogonal. This explains how the reconstruction errors in the zero DC

Author email:dion.tzamarias@uab.cat This work was supported in part by the Spanish Ministry of Economy and Competitiveness (MINECO) and by the European Regional Development Fund (FEDER) under grant RTI2018-095287-B-I00, and by the Catalan Government under grant 2017SGR-463.

GraphBior concentrate near strong edges, while the non-zero DC GraphBior penalizes uniformly each pixel. In the zero DC case, the error near strong edges is given lower weight (because the degree is smaller as depicted in Fig. 1c), therefore reconstruction errors can be greater in those areas, and lower elsewhere.

The authors in [10] generalize the admissibility condition of the fundamental matrices that can be used in GraphBior, leading to biorthogonal graph filterbanks that are approximately orthogonal with respect to different inner products. In this paper we propose a novel graph variation operator $\mathbf{M_K} = \mathbf{K}^{-1/2}\mathbf{L}\mathbf{K}^{-1/2}$, where $\mathbf{K}$ is a diagonal matrix with positive entries, and $\mathbf{L}$ is the combinatorial Laplacian. We exploit the admissibility condition in [10] to construct a fundamental matrix that can replace the normalized or random walk Laplacians in GraphBior. By appropriately optimizing the diagonal modification matrix $\mathbf{K}$, the resulting GraphBior filterbanks are approximately orthogonal with respect to a new inner product. Such an inner product and the proposed variation operator aim to balance the trade-off between energy compaction and degree-error distribution. Several experiments are conducted demonstrating this effect, in the context of image compression, by reporting an average increase in PSNR of up to 0.6 dB at low degree nodes for a small penalty in total PSNR and rate.

This paper is structured as follows. In Section 2, we review the GraphBior filterbanks. In Section 3 we introduced our proposed approach. Sections 4 and 5 provide our main experimental results and conclusions, respectively.

2. BIORTHOGONAL GRAPH FILTERBANKS

2.1. Graph frequency definition

An undirected graph $G = (V, E)$ is composed of a set of vertices V of cardinality $|V| = N$ that are connected through a set of edges E. The weighted graph edges are described through the symmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ with elements a_{ij} referring to the edge weight that connects node i to node j. The diagonal degree matrix $\mathbf{D} = \mathrm{diag}(d_1, d_2, ..., d_N)$ describes the sum of the weighted edges incident to each node, with $d_j = \sum_{i\in V} a_{ji}$. Through a graph variation operator $\mathbf{M}$, one can measure the frequency content of a signal $f \in \mathbb{R}^N$ on the graph. Two well known graph variation operators are the combinatorial Laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{A}$, and the normalized Laplacian, $\boldsymbol{\mathcal{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$.

For a positive definite matrix $\mathbf{Q}$ we define the $\mathbf{Q}$ inner product as $\langle \mathbf{x}, \mathbf{y}, \rangle_{\mathbf{Q}} = \mathbf{y}^\top \mathbf{Q}\mathbf{x}$, and the $\mathbf{Q}$-norm of $\mathbf{x}$ as $\|\mathbf{x}\|_{\mathbf{Q}} = \sqrt{\langle \mathbf{x}, \mathbf{y}, \rangle_{\mathbf{Q}}}$. Given a graph variation operator $\mathbf{M}$ and a Hermitian positive definite matrix $\mathbf{Q}$, the $(\mathbf{M}, \mathbf{Q})$-Graph Fourier Transform (GFT) is defined as $\mathbf{F} = \mathbf{U}^\top \mathbf{Q}$ and its inverse as $\mathbf{F}^{-1} = \mathbf{U}$, where $\mathbf{U}$ is the generalized eigenvector matrix (whose columns are the $(\mathbf{M}, \mathbf{Q})$-Graph Fourier modes) of the fundamental matrix $\mathbf{Z} = \mathbf{Q}^{-1}\mathbf{M}$ [12]. The eigenvalues of the fundamental matrix correspond the the frequency of each graph Fourier mode. It is important to note that the $(\mathbf{M}, \mathbf{Q})$-graph Fourier modes form an orthonormal basis with respect to the $\mathbf{Q}$-inner product.

2.2. Graph Filterbanks

The GraphBior transform can be seen as a design based on the $(\boldsymbol{\mathcal{L}}, \mathbf{I})$-GFT, where filters are polynomials of the normalized Laplacian $\boldsymbol{\mathcal{L}}$. In [10], it is shown that the normalization procedure that is carried out in the zero-DC GraphBior is equivalent to replacing $\boldsymbol{\mathcal{L}}$ by the random walk Laplacian $\mathbf{L}_{RW} = \mathbf{D}^{-1}\mathbf{L}$ as the fundamental matrix to construct the filters. Thus, the $(\boldsymbol{\mathcal{L}}, \mathbf{I})$-GFT is replaced by the $(\mathbf{L}, \mathbf{D})$-GFT. The DC signal corresponds then to the eigenvector of $\mathbf{L}_{RW}$ with the smallest eigenvalue $\lambda_0 = 0$, so that the random walk Laplacian GraphBior has the zero-DC property.

For GraphBior to achieve perfect reconstruction (PR), the graph Fourier modes must obey the spectral folding property [9]. Recently, the spectral folding property has been generalized to GFTs beyond $(\boldsymbol{\mathcal{L}}, \mathbf{I})$ and $(\mathbf{L}, \mathbf{D})$-GFTs, allowing new fundamental matrices other than $\boldsymbol{\mathcal{L}}$ and $\mathbf{L}_{RW}$ to be used in GraphBior [10]. Specifically, for a given bipartite graph and variation operator $\mathbf{M}$ a general admissible GFT with spectral folding can be designed for a choice of $\mathbf{Q}$ according to the following theorem [10]:

Theorem 1. *Given a positive semidefinite variation operator* $\mathbf{M}$ *and a positive definite inner product matrix* $\mathbf{Q}$*, defined on a bipartite graph, the spectral folding condition holds for the modes of the* $(\mathbf{M}, \mathbf{Q})$*-GFT if* $\mathbf{Q} = \mathrm{diag}(\mathrm{diag}(\mathbf{M}))$.

Thus, we are free to choose any variation operator to compute the GraphBior filters, as long as the definition of the $\mathbf{Q}$ inner product follows the theorem. As a result many new GraphBior variations can be introduced besides the non-zero and zero-DC types.

3. PROPOSED DESIGN

3.1. Z-GraphBior

We call $\mathbf{Z}$-GraphBior, a biorthogonal graph filterbank whose filters are computed through polynomials of the $\mathbf{Z} = \mathbf{Q}^{-1}\mathbf{M}$ fundamental matrix, when $\mathbf{M}$ and $\mathbf{Q}$ are chosen according to Theorem 1. When an admissible $(\mathbf{M}, \mathbf{Q})$-GFT is used in GraphBior, the corresponding analysis and synthesis filters are approximately orthogonal with respect to the $\mathbf{Q}$-inner product [10]. A linear operator $\mathbf{T}$ is $\mathbf{Q}$-orthogonal when it preserves the $\mathbf{Q}$-norm of signals, that is $||\mathbf{T}\mathbf{x}||_{\mathbf{Q}} = ||\mathbf{x}||_{\mathbf{Q}}$. In the case of GraphBior the analysis and synthesis filters are approximately $\mathbf{Q}$-orthogonal, meaning that $||\mathbf{T}\mathbf{x}||_{\mathbf{Q}} \approx ||\mathbf{x}||_{\mathbf{Q}}$. The zero-DC GraphBior uses the $(\mathbf{L}, \mathbf{D})$-GFT and therefore the analysis and synthesis filters are approximately orthogonal with respect to the $\mathbf{D}$-inner product. This observation justifies the large accumulation of errors at nodes with low degree when the zero-DC GraphBior is used in image compression applications.

In order to rectify this unwanted behavior of the zero-DC GraphBior, we would like to choose an admissible $(\mathbf{M}, \mathbf{Q})$-GFT whose inner product is close to the one of $(\boldsymbol{\mathcal{L}}, \mathbf{I})$-GFT (the identity matrix) without diverging too much from the zero-DC property of the $(\mathbf{L}, \mathbf{D})$-GFT. We propose the variation operator

$$\mathbf{M}_K = \mathbf{K}^{-1/2}\mathbf{L}\mathbf{K}^{-1/2} = \mathbf{K}^{-1/2}\mathbf{D}\mathbf{K}^{-1/2} - \mathbf{K}^{-1/2}\mathbf{A}\mathbf{K}^{-1/2},$$

where $\mathbf{K}$ is a diagonal positive definite matrix. Following Theorem 1, its admissible inner product matrix is

$$\mathbf{Q}_K = \mathrm{diag}(\mathrm{diag}(\mathbf{M})) = \mathbf{K}^{-1/2}\mathbf{D}\mathbf{K}^{-1/2}.$$

Thus the proposed fundamental matrix that can be used in place of $\boldsymbol{\mathcal{L}}$ or $\mathbf{L}_{RW}$ in GraphBior is

$$\mathbf{Z}_K = \mathbf{I} - \mathbf{K}^{1/2}\mathbf{D}^{-1}\mathbf{A}\mathbf{K}^{-1/2}.$$

Note that $\mathbf{K}^{1/2}\mathbb{1}$ is the lowest frequency graph Fourier mode. By modifying $\mathbf{K}$ our proposed $(\mathbf{M}_K, \mathbf{Q}_K)$-GFT achieves a tradeoff between $(\boldsymbol{\mathcal{L}}, \mathbf{I})$ and $(\mathbf{L}, \mathbf{D})$-GFTs. Thus, if $\mathbf{K} \approx \mathbf{D}$ our proposed variation operator resembles the normalized Laplacian leading to the $(\boldsymbol{\mathcal{L}}, \mathbf{I})$-GFT. On the other hand when $\mathbf{K} \approx \mathbf{I}$, the variation operator is closer to the Combinatorial Laplacian leading to the $(\mathbf{L}, \mathbf{D})$-GFT.

3.2. Optimization of K

In order to improve the tradeoff between the $(\boldsymbol{\mathcal{L}}, \mathbf{I})$ and $(\mathbf{L}, \mathbf{D})$-GFTs we define an optimization function $C(\mathbf{k})$. The vector $\mathbf{k}_0$ that minimizes $C(\mathbf{k})$ will be used to define the desired $(\mathbf{M}_K, \mathbf{Q}_K)$-GFT and therefore $\mathbf{Z}_K$. Note that we assume that the graph structure and the combinatorial Laplacian are known, and we only seek to learn the vector $\mathbf{k}$. Our objective function $C(\mathbf{k})$ is defined as:

$$C(\mathbf{k}) = (\mathbf{k}^\top)^{1/2}\mathbf{L}\mathbf{k}^{1/2} + \alpha\|\mathbf{D}\mathbf{k}^{-1} - \mathbb{1}\|_2^2, \qquad (1)$$

where the non negative $\alpha \in R^+$ is a learning parameter, and $\mathbf{k} = \text{diag}(\mathbf{K})$. By adjusting the value of α we are able to control how close to $\boldsymbol{\mathcal{L}}$ or $\mathbf{L}_{RW}$ our fundamental matrix $\mathbf{Z}_K$ will become.

The first term in (1), $(\mathbf{k}^\top)^{1/2}\mathbf{L}\mathbf{k}^{1/2}$, is the variation of $\mathbf{k}^{1/2}$ with respect to $\mathbf{L}$ (i.e., the Laplacian quadratic form of $\mathbf{k}^{1/2}$). Since $\mathbf{k}^{1/2}$ corresponds to the lowest frequency eigenvector for the $(\mathbf{M}_K, \mathbf{Q}_K)$-GFT, and given that $\mathbb{1}^\top\mathbf{L}\mathbb{1} = 0$, this first term favors solutions with the lowest frequency for $(\mathbf{M}_K, \mathbf{Q}_K)$-GFT close to the DC signal. This is justified because, as noted in our discussion of the $\mathbf{L}_{RW}$-GraphBior vs $\boldsymbol{\mathcal{L}}$-GraphBior trade-off, zero frequency close to $\mathbb{1}$ leads to better energy compaction for typical images.

The second term in (1), expresses the squared Euclidean distance between the diagonal elements of the identity and the $\mathbf{Q}_K$-inner product matrix. Minimizing this distance allows $\mathbf{Q}_K$ to approach $\mathbf{I}$. This translates in the filters of the $\mathbf{Z}_K$-GraphBior to become close to orthogonal with respect to the identity-inner product, rather than the degree-inner product. Consequentially the $\mathbf{Z}_K$-GraphBior alleviates the degree-sensitive distribution of errors that hinders the $\mathbf{L}_{RW}$-GraphBior. By incorporating those two terms in an optimization problem we are able to calculate the desired matrix $\mathbf{K}$ such that the resulting fundamental matrix $\mathbf{Z}_K$ achieves a tradeoff controlled by α between desired properties of $\mathbf{L}_{RW}$ and $\boldsymbol{\mathcal{L}}$.

To solve the proposed optimization problem we apply a change of variable $\mathbf{x} = \mathbf{k}^{1/2}$ and transform the cost function to:

$$C(\mathbf{x}) = \mathbf{x}^T\mathbf{L}\mathbf{x} + \alpha\|\mathbf{D}\mathbf{x}^{-2} - \mathbb{1}\|_2^2.$$

The convexity of $C(\mathbf{x})$ is guaranteed if the Hessian of the second term, $H(\|\mathbf{D}\mathbf{x}^{-2} - \mathbb{1}\|_2^2)$, is positive semidefinite, since $\mathbf{x}^T\mathbf{L}\mathbf{x}$ is convex. Given that $\mathbf{d} = \mathbf{D}\mathbb{1}$ as well as $\mathbf{x}$ are composed by positive elements, by imposing the restriction that $\mathbf{x} \leq \sqrt{(5/3)\mathbf{d}}$ we assure that all the elements of the diagonal matrix $H(\|\mathbf{D}\mathbf{x}^{-2} - \mathbb{1}\|_2^2)$ are non negative. Thus the convex optimization problem that we propose for the computation of $\mathbf{x}$ becomes:

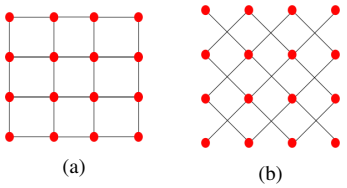
$$\arg\min_{\mathbf{x}} C(\mathbf{x}),\ s.t.\ 0 < \mathbf{x} \leq \sqrt{(5/3)\mathbf{d}}.$$

4. COMPRESSION EXPERIMENTS

In this section we describe the GraphBior image compression scheme. Next, we present the experimental settings and our numerical results.

4.1. Compression scheme

We follow the compression scheme presented in [5], using a 2D separable 2-channel implementation of GraphBior. The two bipartite graphs that are used can be seen in Figures 2a and 2b, connecting nodes in a vertical-horizontal or diagonal manner respectively. In order for the graph structures to be efficiently transmitted to the decoder we adopt an edge-aware graph design [11] such that non-zero edges of the graphs take one of two possible values.



(a) (b)

Fig. 2: Bipartite graphs connecting nodes through: (a) vertical-horizontal edges, and (b) diagonal edges

Image	Experiment	α_1	α_2	min. weight
peppers	1	2.86	$4 \cdot 10^{-4}$	0.1
Barbara	1	4.32	$5.4 \cdot 10^{3}$	0.23
peppers	2	3.48	$3.48 \cdot 10^{-8}$	10^{-3}

Table 1: Table of parameters used in the first and second experiments.Fundamental matrices $\mathbf{Z}_{K1}$and $\mathbf{Z}_{K2}$ are computed through learning parameters α_1 and α_2 respectively. The edge-aware graph uses as a minimum weight the values indicated in the last column.

The high valued edge weight is equal to 1 whereas a lower value is assigned to weaker edges. The low value weight is reserved for pixels that are connected across high gradient areas of the image. These areas are found through the Sobel edge detection algorithm. Then the pairs of nodes connected through low weighted edges can be expressed as image contours which are then encoded through an arithmetic edge encoder [13]. In all of our experiments we use 5-levels of GraphBior and the quantized analysis coefficients are then encoded through a standard adaptive arithmetic encoder.

The complexity of the optimization procedure is dominated by the computation of $\mathbf{L}\mathbf{x}$. For the case of 4-connected graphs of N nodes, like the ones used in this paper, the Laplacian matrix is sparse with $O(N)$ non-zero elements [14]. Therefore, since the tree filter-bank has at most $O(log(N))$ levels we can deduce that the overall complexity of the optimization problem is $O(N)$.

4.2. Results

Based on the aforementioned compression scheme we compare the results obtained using the proposed fundamental matrix $\mathbf{Z}_K$ for various learning parameters α to those achieved with $\boldsymbol{\mathcal{L}}$ and $\mathbf{L}_{RW}$. An advantage of the $\mathbf{Z}_K$-GraphBior is that we control the trade-off between a zero-DC response and a uniform error distribution. To study this trade-off, we measure both the total reconstruction distortion and the distortion at low degree nodes (corresponding to high gradient pixels). In the following experiments we create the graphs and calculate the fundamental matrices $\mathbf{Z}_K$ using the empirically chosen parameters according to Table 1.

In our first experiment we compare the $\boldsymbol{\mathcal{L}}$, $\mathbf{L}_{RW}$ and $\mathbf{Z}_K$-GraphBiors through the Bjontegaard metric [15]. We use two versions of our proposed fundamental matrix, $\mathbf{Z}_{K1}$ and $\mathbf{Z}_{K2}$, each one for a different learning parameter α. Specifically, $\mathbf{Z}_{K1}$ has been computed by choosing $\alpha = \alpha_1$ such that the the norm term in (1) is penalized the most. This results in $\mathbf{Q}_{K1}$ resembling more to the identity matrix, rather than the degree matrix. Thus we expect the $\mathbf{Z}_{K1}$-GraphBior to penalize less harshly the low degree nodes for the cost of an overall lower energy compaction. Conversely by penalizing more the Laplacian quadratic term of $C(x)$ using $\alpha = \alpha_2 < \alpha_1$, the $\mathbf{Z}_{K2}$-GraphBior achieves better energy compaction.

Peppers	$\mathcal{L}$	$\mathbf{Z}_{K1}$	$\mathbf{Z}_{K2}$	$\mathbf{L}_{RW}$
Total PSNR	-2.005	-0.282	-0.024	**0**
Edge PSNR	-0.509	**0.614**	0.161	0
Rate(%)	36.51	4.854	0.377	**0**

Barbara	$\mathcal{L}$	$\mathbf{Z}_{K1}$	$\mathbf{Z}_{K2}$	$\mathbf{L}_{RW}$
Total PSNR	-1.292	-0.132	-0.028	**0**
Edge PSNR	-0.332	**0.507**	0.115	0
Rate(%)	11.317	1.158	0.245	**0**

Table 2: Table of Bjontegaard measurements comparing the Graph-Bior variations using $\mathcal{L}$ and the proposed $\mathbf{Z}_{K1}$, $\mathbf{Z}_{K2}$ against $\mathbf{L}_{RW}$. The comparisons are done using images *peppers* and *Barbara*

For each GraphBior variation we carry out a series of measurements by compressing the images *peppers* and *Barbara* for multiple bitrates. At each measurement, besides the bitrate, we also measure the total PSNR of the reconstructed image as well as its edge-PSNR. By edge-PSNR we refer to the distortion on edge pixels, i.e., pixels having at least 2 weak edge connections in at least one of the two bipartite graphs.

In Table 2 we report the Bjontegaard measurements comparing the $\mathcal{L}$, $\mathbf{Z}_{K1}$ and $\mathbf{Z}_{K2}$-GraphBior against the $\mathbf{L}_{RW}$-GraphBior. In the first line we compare the difference in PSNR for all the pixels of the image, the second line the edge-PSNR and the last line the difference percentage in rate.

We observe that the best performing GraphBior, in terms of both rate and total PSNR, uses the random walk Laplacian $\mathbf{L}_{RW}$. This is expected since it has a superior energy compaction due to its zero-DC property. However, when comparing the edge-distortion we note that it is $\mathbf{Z}_{K1}$ that performs the best. We also note that since $\mathbf{Z}_{K2}$ is closer to $\mathbf{L}_{RW}$ it outperforms $\mathbf{Z}_{K1}$ at rate and total PSNR but not edge-PSNR. Furthermore the difference between $\mathbf{Z}_{K2}$ and $\mathbf{L}_{RW}$ is very small when regarding the bitrate and total distortion. The rate distortion plot of these GraphBior variations are depicted in Figure 3.

The next experiment compares the distribution of errors as a function of the degree of each node. For each fundamental matrix we compress *peppers* such that the reconstructed image has a distortion of approximately 35.9dB. Obviously the bitrate of each of those compressed images is not the same. In Figures 4a and 4b we display the average of the absolute value of the errors of the nodes in function of their degree for the diagonal-edge and vertical-horizontal-edge graphs respectively. For the edge-aware graph design, the minimum edge weight has been set to 10^{-3}. Thus the low-edge nodes have a degree that is lower or equal to 2.002. We can clearly see that low-degree nodes are best reconstructed by the $\mathcal{L}$-GraphBior whereas the $\mathbf{L}_{RW}$-GraphBior distorts them the most. For higher degree nodes, the difference in average error between the zero and non zero-DC GraphBiors becomes smaller. Meanwhile the proposed $\mathbf{Z}_{K2}$-GraphBior and (even more so) the $\mathbf{Z}_{K1}$-GraphBior penalize less the low degree nodes when compared to $\mathbf{L}_{RW}$-GraphBior.

5. CONCLUSION

It is known that the main disadvantage of the $\mathbf{L}_{RW}$-GraphBior used in image compression, is related to the accumulation of large errors at low degree nodes. Though, due its zero-DC response, the $\mathbf{L}_{RW}$-GraphBior is linked to a large energy compaction property, which is very useful in image compression. In this paper we propose a new

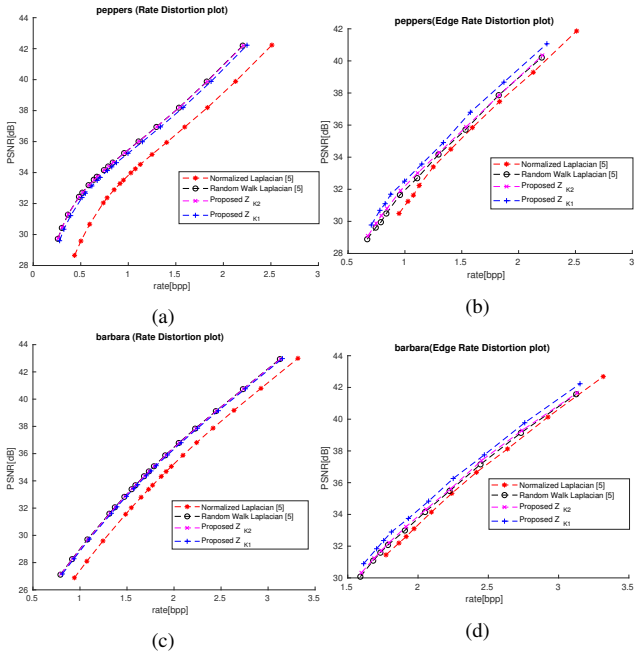


Fig. 3: Rate Distortion curves for *Peppers* (a) Total Distortion, (b) Edge-Distortion and *Barbara* (c) Total Distortion and (d) Edge-Distortion

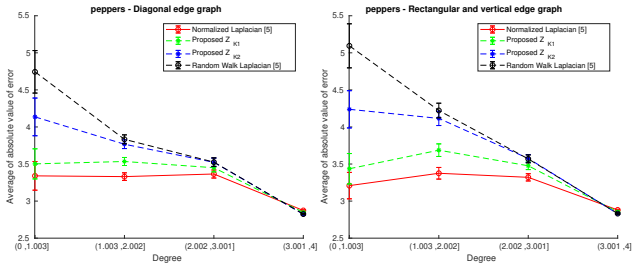


Fig. 4: Average absolute value of node error with a 95% confidence interval versus degree for (a) Diagonal-edge graph (b) vertical-horizontal-edge graph. The *peppers* image was used and each reconstruction had a total PSNR of approximately 35.9dB

variation operator as well as an inner-product matrix which depend on a diagonal positive definite matrix $\mathbf{K}$. The proposed variation operator and the inner-product matrix are such that the resulting fundamental matrix $\mathbf{Z}_K$ obeys to the spectral folding property required from GraphBior. By defining a learning parameter α and solving a convex optimization problem we learn the matrix $\mathbf{K}$, such that the $\mathbf{Z}_K$-GraphBior penalizes in more evenly manner low and high degree nodes without deviating too much from the zero-DC characteristic of the $\mathbf{L}_{RW}$-GraphBior. In our experimental results we show that by choosing the appropriate learning parameter the resulting $\mathbf{Z}_K$-GraphBior outperforms the $\mathbf{L}_{RW}$-GraphBior when comparing the distortion of low degree nodes without penalizing much the global distortion. Our design can find useful applications in image compression when one might require less distortion on high gradient areas of the image.

6. REFERENCES

[1] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[3] Gene Cheung, Enrico Magli, Yuichi Tanaka, and Michael K Ng, "Graph spectral image processing," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 907–930, 2018.

[4] David I Shuman, "Localized spectral graph filter frames: A unifying framework, survey of design considerations, and numerical comparison," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 43–63, 2020.

[5] Sunil K Narang, Yung-Hsuan Chao, and Antonio Ortega, "Critically sampled graph-based wavelet transforms for image coding," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 2013, pp. 1–4.

[6] Akie Sakiyama, Kana Watanabe, Yuichi Tanaka, and Antonio Ortega, "Two-channel critically sampled graph filter banks with spectral domain sampling," *IEEE Transactions on Signal Processing*, vol. 67, no. 6, pp. 1447–1460, 2019.

[7] Jin Zeng, Gene Cheung, Yung-Hsuan Chao, Ian Blanes, Joan Serra-Sagristà, and Antonio Ortega, "Hyperspectral image coding using graph wavelets," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1672–1676.

[8] Sunil K Narang and Antonio Ortega, "Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs," *IEEE transactions on signal processing*, vol. 61, no. 19, pp. 4673–4685, 2013.

[9] Sunil K Narang and Antonio Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2786–2799, 2012.

[10] Eduardo Pavez, Benjamin Girault, Antonio Ortega, and Philip A Chou, "Spectral folding and two-channel filter-banks on arbitrary graphs," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[11] Sunil K Narang, Yung Hsuan Chao, and Antonio Ortega, "Graph-wavelet filterbanks for edge-aware image processing," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2012, pp. 141–144.

[12] Benjamin Girault, Antonio Ortega, and Shrikanth S Narayanan, "Irregularity-aware graph fourier transforms," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5746–5761, 2018.

[13] Ismael Daribo, Gene Cheung, and Dinei Florencio, "Arithmetic edge coding for arbitrarily shaped sub-block motion prediction in depth video compression," in *2012 19th IEEE International Conference on Image Processing*. IEEE, 2012, pp. 1541–1544.

[14] Eduardo Pavez, Hilmi E Egilmez, Yongzhe Wang, and Antonio Ortega, "GTT: Graph template transforms with applications to image coding," in *2015 Picture Coding Symposium (PCS)*. IEEE, 2015, pp. 199–203.

[15] Gisle Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.

Chapter 4

Compression of point cloud geometry through a single projection

CORE quality factor: A+

Compression of point cloud geometry through a single projection

Dion E.O. Tzamarias*, Kevin Chow*, Ian Blanes* and Joan Serra-Sagristà*

*Department of Information and Communications Engineering,
Universitat Autònoma de Barcelona, Carrer de les Sitges s/n, Campus UAB,
Cerdanyola del Vallès, Barcelona, 08193, Spain
dion.tzamarias@uab.cat, kevin.chow@uab.cat
ian.blanes@uab.cat, joan.serra@uab.cat

Abstract

Point cloud data have been put under the spotlight by many applications that play an increasingly important role in our every day lives. Their large size and ever-growing prevalent use cases have raised the interest in specialized compression algorithms for point cloud data. In this paper we propose a lossless intra-frame encoder for point cloud geometry. It relies on a single projection of the entire point cloud on a predetermined plane, combined with a context-adaptive binary arithmetic encoder. Our approach simplifies the current best performing approach for intra-frame compression. The experimental results indicate that our proposal not only improves the performance of all other intra-frame approaches, but it even surpasses the performance of state-of-the-art inter-frame approaches. Furthermore, we suggest to replace the adaptive encoder with a semi-adaptive approach for further performance gains.

Introduction

Point cloud data permit the digital representation of objects and sceneries in their three-dimensional form through registering the coordinates and attributes of a collection of points. Advances in 3D scanning technologies have facilitated the acquisition of such data, making way for numerous applications of emerging importance. Some applications utilizing point clouds are Virtual/Augmented-Reality, automated driving, telecommunications, world heritage preservation, etcetera [1].

In order to facilitate the processing of point clouds, a quantization procedure of the point coordinates often follows after their acquisition, mapping each point to an occupied unitary volume in a regular cubic grid. Each unitary volume of the cubic grid is called *voxel*, whereas the quantized point clouds are referred to as *voxelized*.

Typically, point clouds may consist of hundreds of thousands of points. Given this huge size, compression of such data is of paramount importance to enable the aforementioned applications. The compression of parameters such as colour, reflectibility or normals of the points is accomplished through *attribute compression*, whereas the compression of the coordinates of the points is referred to as *geometry compression*. As opposed to *static* point clouds, the *dynamic* point clouds vary over time. Therefore, similarly to 2D video encoders, dynamic point clouds can be compressed by inter or intra coders depending on whether they do or do not exploit temporal redundancies. Moreover, analogously to video encoders, the attribute or geometry compression of point clouds can be achieved in a lossy or a lossless manner.

2021 Data Compression Conference (DCC) | 978-1-6654-0333-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/DCC50243.2021.00014

2375-0359/21/$31.00 ©2021 IEEE
DOI 10.1109/DCC50243.2021.00014

In this paper we propose an intra frame lossless geometry compression algorithm for voxelized point clouds. We do so by improving over the currently best performing intra frame encoder [2], exploiting the tradeoff between the transmission of silhouettes and the cost of encoding non-occupied voxels. We then report experimental results for the geometry compression of point cloud sequences, showing that the proposed method outperforms state-of-the-art lossless geometry compression approaches.

The organization of the paper is as follows. In the next section we provide a description of state-of-the-art geometry compression algorithms for voxelized point clouds. In the third section we describe the design of the proposed encoder and provide experimental results, illustrating the competitive performance of our approach. In the fourth section we provide a detailed analysis of on-going studies, unveiling possible gains related to the use of semi-adaptive arithmetic encoders. Finally, we bring forward our conclusions in the last section.

State of the art

An increase in the popularity of point cloud based applications has triggered the development of many different algorithms that enable the lossless geometry compression of this particular type of data.

One of the most common approaches for geometry compression relies on the octree decomposition [3]. An octree is created by confining the voxelized point cloud in a single cube, which is then iteratively subdivided into 8 smaller equally sized cubes called octants. Each octant is further subdivided, under the condition of it containing at least one occupied voxel. Therefore, each level of decomposition is characterized as a byte with 1s at the locations that contain occupied voxels.

By combining entropy encoders with the octree representation, the compression ratio of such lossless methods improves significantly [4]. One such example introduces arithmetic or LZW coders, with contexts relating to the parents of each octant [5]. An improvement of the prior method was proposed in [6], which also included an inter-frame variant and used a super-resolution technique similar to [7]. Other inter-frame approaches change the scanning order in the octree using a reference frame [8] or exploit the temporal redundancies of point cloud frames through the XOR operator [9].

The MPEG standard for intra-frame geometry compression, usually referred to as G-PCC, is also based on the entropy encoding of an octree representation [1]. On the other hand, the inter-frame variant of MPEG, known as V-PCC, makes use of 3D to 2D projections in order to apply standard video codecs such as HEVC [1]. Another projection approach proposes a polynomial surface fitting algorithm to improve the performance of V-PCC [10].

In addition to the V-PCC encoder, there are many different approaches to the octree representation that also facilitate the geometry compression of point cloud data. In [11], Zhu et al. use a binary tree structure to model the problem of optimally traversing through all occupied voxels as a traveling salesperson problem and use a predictive approach to encode the residual distances between the coordinates of

successive points. Other geometry encoders are based on cellular automata block transforms where each voxel is seen as an automaton whose state evolves in function of the values of neighboring voxels [12], [13].

Recently, a new class of geometry encoders represent the voxelized point cloud using a three dimensional binary matrix $G(x, y, z) \in \mathbb{Z}_2^{N \times N \times N}$, called occupancy array. Elements of the occupancy array that share common coordinates with occupied voxels are represented by 1s, whereas the empty voxels are represented as 0s. Each $N \times N$ slice of the 3D matrix that is perpendicular to one of the x, y or z-axis is called an image. The family of aforementioned geometry coders compress the occupancy array by encoding each image through context-adaptive binary arithmetic encoders while the number of non-occupied voxels that are encoded is reduced through various decompositions.

In [14], Rosário and Peixoto encode an occupancy array through a 2D context-adaptive binary arithmetic encoder, similar to JBIG. A reduction of the encoded empty voxels is achieved thanks to their proposed boolean decomposition, performed by a series of logical operations on pairs of adjacent images of the occupancy array.

A dyadic decomposition is later introduced by Peixoto for the compression of the occupancy array [2]. In this decomposition, the occupancy array is iteratively sliced into two equally sized subvolumes along a predetermined axis. Each time a new subvolume is generated, an $N \times N$ binary image –called silhouette– is computed by projecting each subvolume on the plain perpendicular to the slicing axis. This way, a series of parent and children silhouettes is defined. The purpose of the silhouettes is to signal, with increasing precision, the location of non-occupied voxels for a group of adjacent images. Notice that a value of 0 at a location of the silhouette means that none of the images linked to the silhouette has a non-empty voxel at that specific location. Through logical operations between parent and children silhouettes, the dyadic decomposition reduces the number of 1s of each silhouette that need to be transmitted to the decoder. By additionally introducing 3D contexts, this technique improves the results of the boolean decomposition and, to the best of our knowledge, contributes the most competitive results regarding intra-frame geometry encoders. An inter-frame variant using the dyadic decomposition has been recently proposed [15], employing 4D contexts, i.e., also exploiting the temporal redundancy; it further improves the performance of geometry encoding.

Single-silhouette projection

Our proposed lossless intra-frame geometry compression algorithm for point cloud data is inspired by the intra-frame geometry encoder using the dyadic decomposition [2]. The performance of occupancy array-type encoders is based on the tradeoff between increasing side information and reducing the number of non-occupied voxels that are transmitted to the decoder. As mentioned in the preceding section, the reduction of the transmitted non-occupied voxels can be accomplished through various decompositions. In the case of the dyadic decomposition, this is achieved through a series of silhouettes that signal, with increasing precision, the location of non-occupied voxels to the decoder.

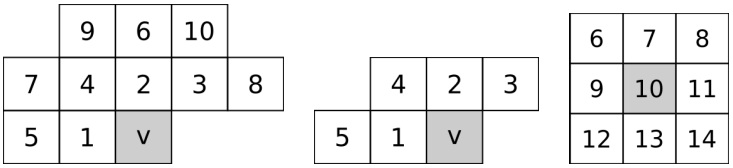


Figure 1: Contexts used to encode pixel v. In (a), 2D contexts used on the projection matrix. In (b) and (c), voxels of, respectively, the current image and the previous image form the 3D contexts.

Figure 2: Depiction of single frames from the Microsoft upper body database from the sequences (a) Andrew, (b) David, (c) Phil, (d) Ricardo and (e) Sarah.

As we will show, the act of signaling very accurately non-occupied voxels does not compensate the cost of transmitting the large number of necessary silhouettes as side-information. Therefore, we propose a single projection along the plane perpendicular to an axis of our choice.

In our proposed method, we adopt the occupancy-array representation, $G(x, y, z) \in \mathbb{Z}_2^{N\times N\times N}$, of the voxelized point cloud for its intra-frame geometry compression. We use $G_m^{(k)}$ to denote the k^{th} $N \times N$ image of $G(x, y, z)$ perpendicular to axis m.

The first step of our proposed algorithm is to project the entire occupancy array $G(x, y, z)$ on one of the 2D planes perpendicular to the x, y or z-axis. The $N \times N$ binary matrix resulting from the projection of $G(x, y, z)$ onto plane m is referred to as the projection matrix P_m and is computed by the following logical sum:

$$P_m(i, j) = \sum_k G_m^{(k)}(i, j).$$

The projection matrix P_m is identical to the first silhouette that is calculated in [2]. It should be noted that we use the context-adaptive binary arithmetic encoder (CABAC) and contexts in [2] for the compression of both the projection matrix and the images of the occupancy array.

The projection matrix is encoded using 10 pixels as contexts, as depicted in Figure 1(a). Once the projection matrix is encoded, we deploy the single-mode encoding procedure of [2]. For each image, the only voxels that are being encoded are the ones that share the same 2D coordinates with the elements of P_m that are equal to 1. Each voxel of an image is then encoded with CABAC using 3D contexts. Here 3D contexts

(a) (b) (c) (d)

Figure 3: Depiction of single frames from the full body JPEG Pleno database from the sequences (a) Longdress, (b) Loot, (c) RedAndBlack and (d) Soldier.

mean using 5 pixels as contexts from the same image –Figure 1(b)– and 9 pixels as contexts from the previously encoded image –Figure 1(c).

By using a single projection matrix we dramatically decrease the side information that needs to be transmitted to the decoder in the form of silhouettes. Furthermore, we simplify the algorithm of [2], since the entire dyadic decomposition is replaced by a single projection.

In order to compare our proposal with the state of the art, we use the point cloud sequences from the Microsoft Voxelized Upper Bodies dataset [16] as well as the 8i Voxelized Full Bodies from the JPEG Pleno Database [17], with 9 and 10 bit spatial resolution, respectively. Illustrations of such point cloud sequences can be seen in Figure 2 and Figure 3. All methods are tested on the first 100 frames of each point cloud sequence. It should be noted that we do not search for the optimal projection plane. Instead, we follow an empirically-derived approach by projecting the occupancy array always on the plane perpendicular to the z-axis, unless the spread of the points is much larger along the y-axis. In the latter case, the projection of the occupancy array is performed on the plane perpendicular to the y-axis.

We compare our methods using the P_z and P_y projection matrices against the plain octree representation, MPEG G-PCC v7.0 variant [18]. We also compare against the intra- and inter-geometry compression algorithms that use the dyadic decomposition [2] and [15], which are denoted as S-3D and S-4D, respectively. The rate at which the geometry of a point cloud is losslessly encoded is measured in bits per occupied voxel (bpov). The results of the comparisons are reported in Table 1.

As one can see, our proposed method outperforms both the intra-frame and the inter-frame variants of the dyadic decomposition, so far the best performing approaches. Our proposal does not select the optimal projection plane, in contrast to the S-3D method that utilizes a computationally expensive trial-and-error approach to guarantee that the decomposition is done along the optimal axis.

In the case of the Microsoft Upper Bodies dataset, we can see an average gain of 11.29% over G-PCC, of 9.01% against S-3D, and of 7.12% over the inter-frame S-4D approach. Moreover, our proposal is overall the best performing one on the Full Body JPEG Pleno dataset, reporting average gains of 13.27% over G-PCC, of 6.76% over S-3D, and of 2.17% over S-4D. In addition, our proposal easily outperforms the plain

Table 1: Compression between our proposed methods using the P_z or P_y projection matrix against a plain Octree encoding, G-PCC [18], S-3D [2] and S-4D [15]. We report the average rate in bpov for the lossless geometry compression of the 100 first frames of sequences from the Microsoft Voxelized Upper Bodies dataset [16] and the Full Body JPEG Pleno dataset [17].

	Intra Coders					Inter Coder	Gains of **P** over		
Sequence	**Octree**	**G-PCC**	**S-3D**	**Proposed**		**S-4D**	**G-PCC**	**S-3D**	**S-4D**
				Pz	**Py**				
andrew9	2.58	1.14	1.12	**1.02**	1.05	1.08	-10.53%	-8.93%	-5.56%
david9	2.62	1.07	1.06	**0.96**	1.02	1.05	-10.28%	-9.43%	-8.57%
phil9	2.64	1.18	1.14	**1.03**	1.05	1.13	-12.71%	-9.65%	-8.85%
ricardo9	2.59	1.10	1.04	**0.96**	0.97	1.02	-12.73%	-7.69%	-5.88%
sarah9	2.61	1.08	1.07	**0.97**	1.01	1.04	-10.19%	-9.35%	-6.73%
Average	2.61	1.11	1.09	**0.99**	1.02	1.06	-11.29%	-9.01%	-7.12%
longdress	2.99	1.03	0.95	0.94	**0.89**	0.95	-13.59%	-6.32%	-6.32%
loot	2.98	0.97	0.92	0.91	**0.85**	0.91	-12.37%	-7.61%	-6.59%
redandblack	3.00	1.10	1.02	1.00	**0.95**	1.02	-13.64%	-6.86%	-6.86%
soldier	3.00	1.04	0.96	0.93	0.90	**0.81**	-13.46%	-6.25%	11.11%
Average	2.99	1.04	0.96	0.95	**0.90**	0.92	-13.27%	-6.76%	-2.17%

octree representation in both upper body and full body datasets.

As we have previously mentioned, a projection along the z-axis is not always the best choice. For example, in our proposed method, our empirically derived axis selection rule favors a projection along the y-axis for the sequences of the full body dataset. This fluctuation in performance might be related to the large difference in the range of coordinates of the points along the z-axis compared to the y or x-axis. Therefore, according to our axis selection rule, in the cases where the variance of the points along the y-axis is sufficiently larger than along the z-axis, we project the occupancy array along the former.

The reason why our proposed scheme outperforms S-3D and S-4D might not be only due to transmitting less silhouettes. It is clear that by using less silhouettes, there is a larger number of empty voxels that are transmitted to the decoder. However, the most frequent context of the additionally transmitted empty voxels is composed entirely of zeros. Therefore, its nature is such that it expresses very accurately the probability of a voxel being void, since it is very rare for an occupied voxel to be surrounded entirely by non-occupied voxels. Hence, the inexpensive encoding cost of the transmitted surplus of non-occupied voxels does not compromise the overall bitrate.

Analysis

In addition to the single-projection proposal, we argue that further gains can be achieved by replacing CABAC by a semi-adaptive variation. The occupancy array-

based geometry compression algorithms [2, 14, 15] encode the transmitted voxel coordinates through a context-adaptive entropy encoder. Said encoders allow to bypass the transmission of the frequency tables that are necessary for the computation of the conditional probabilities, describing the occupancy of voxels for a given context, as well as the frequency of occurrence of each context. This is done by progressively updating the frequency table during the encoding process, gradually approximating in a more accurate manner the true marginal and conditional probabilities. Therefore, the decoder is able to gradually reconstruct the frequency tables without receiving any side information, which is an advantage over non-adaptive entropy encoders.

In some cases though, the updated probabilities converge too slowly to the true ones and, while the estimations are not yet good enough, the voxels are not encoded efficiently. This might occur in the presence of a redundant amount of contexts, which might very well be the case in our proposed method, where 2^{14} contexts are used to encode the pixels of the images. Thus, the transmission of a partial frequency table can offer more gains over a fully-adaptive approach.

We present a study on how the convergence speed of the updated probabilities affects the compression ratio of our proposal. We compute the theoretical average conditional information per occupied voxel through the set of encoded occupied voxels $v_o \in \mathcal{V}_o$, the set of encoded non-occupied voxels $v_n \in \mathcal{V}_n$, and the set of contexts $c \in \mathcal{C}$ as follows:

$$I = -\frac{\sum_{c\in\mathcal{C}}(\sum_{v_o\in\mathcal{V}_o} log_2(p(v_o|c)) + \sum_{v_n\in\mathcal{V}_n} log_2(p(v_n|c)))}{NOV}.$$

The number of occupied voxels is expressed as NOV, whereas $p(v_o|c)$ is the probability of voxel v_o being occupied and $p(v_n|c)$ is the probability of voxel v_n being non-occupied (empty) for a given context c.

If we compute I using the true probabilities, we obtain the conditional entropy multiplied by the total number of encoded voxels and divided by NOV. This value, which we shall call LB, acts as a theoretical lower bound to the bitrate for the given contexts, in the case where the true probabilities are known. The necessary side information for the transmission of the frequency table is obviously not taken into account.

To compute I for the adaptive case, we progressively calculate the sum through updating the occupancy and non-occupancy probabilities $p(v_o|c)$ and $p(v_n|c)$ respectively. This rate, which we symbolize as ALB, is the theoretical lower bound of any adaptive entropy encoder using the contexts $c \in \mathcal{C}$.

Using our proposed method, we measure the average bitrate, in bpov, of the first 100 frames for various point cloud sequences by projecting along the z-axis and disregarding the information related to the transmission of the projection matrix P_z. In Table 2, using our proposed geometry compression algorithm, we display the actual rate (AR) and compare it against the theoretical lower bounds, associated with the true probabilities (LB), and with the adaptive probabilities (ALB).

If we focus on the second and third columns of Table 2, we observe that the actual bitrate resulting from the CABAC is very close to the theoretical minimum for adaptive entropy encoders. However, a significant difference in rate between the

Table 2: Rates for the compression of the images without taking into account the cost of the projection matrix. We compare the actual rate (AR), its theoretical lower bound using adaptive probabilities (ALB) and its theoretical lower bound using the true probabilities (LB). The rates are averages of the first 100 frames for each sequence and the occupancy array is projected along the z-axis. The average side information for encoding 4% of the contexts is displayed under SI.

Sequence	AR	ALB	LB	SI	Gains of $SI + LB$ over AR
Andrew9	1.01	1.00	0.88	0.07	-5.94%
David9	0.94	0.93	0.83	0.06	-5.32%
Phil9	1.02	1.00	0.90	0.06	-5.88%
Ricardo9	0.94	0.93	0.81	0.08	-5.32%
Sarah9	0.95	0.94	0.82	0.07	-6.32%
Loot10	0.90	0.88	0.82	0.03	-5.56%
Longdress10	0.94	0.92	0.86	0.02	-6.38%
RedAndBlack	0.99	0.98	0.90	0.03	-6.06%
Soldier	0.92	0.90	0.85	0.02	-5.43%

theoretical lower bounds is evident, leading us to believe that, indeed, the updated probabilities converge slowly to the true ones, resulting in inefficient encoding of voxels. Therefore, by selecting a few of the most important contexts and by transmitting their corresponding probabilities, we might be able to obtain further gains. In the second to last column of Table 2 we display, in bpov, the side information generated by encoding the probabilities corresponding to the 4% of the total number of contexts. In this scenario, we assume that for each context, the conditional occupancy probability is encoded by a 2-byte word whereas the context identifier is encoded with 14 bits. Assuming that the encoding rate remains close to the true probability lower bound, a semi-adaptive approach could lead to further gains of up to 6.3% with respect to the actual rate.

On the other hand, in Table 3 we display the true rate as well as the theoretical lower bounds corresponding to the encoding of the projection matrix. In this case, the adaptive approach is a good choice for the encoding of the projection matrix given that the differences between the theoretical lower bounds and the true rate are very small.

Based on these results we speculate that using a semi-adaptive entropy encoder to compress the images, by transmitting part of the frequency table, can lead to important gains if we manage to transmit the necessary side information efficiently without deviating significantly from the theoretical lower bound. Furthermore, as we can see from the last column of Table 3, the percentage of transmitted occupied voxels is very small, reinforcing our claim that most empty voxels are inexpensively encoded. We also outline that decompositions that can inexpensively reduce the number of transmitted empty voxels are still very promising for occupancy array-based methods.

Table 3: Rates for the compression of the projection matrix. We compare the actual rate (AR), its theoretical lower bound using adaptive probabilities (ALB) and its theoretical lower bound using the true probabilities (LB).The percentage of transmitted occupied voxels is displayed under TOV. The rates and percentages are averages of the first 100 frames for each sequence and the occupancy array is projected along the z-axis.

Sequence	AR	ALB	LB	% TOV
Andrew9	0.02	0.02	0.01	2.27
David9	0.02	0.02	0.02	1.71
Phil9	0.02	0.02	0.02	1.76
Ricardo9	0.02	0.02	0.01	1.79
Sarah9	0.02	0.02	0.02	2.06
Loot10	0.01	0.01	0.01	1.19
Longdress10	0.01	0.01	0.01	1.44
RedAndBlack	0.01	0.01	0.01	1.59
Soldier	0.01	0.01	0.01	1.26

Conclusion

We present a lossless intra-frame geometry compression algorithm that replaces the dyadic decomposition of [2] with a single projection of the occupancy array along a plane of our choice. Our experimental results show that we outperform the state of the art methods by reporting gains of up to 9.6% and 8.8% over the previously best performing intra and inter-frame geometry compression algorithms, respectively [2] and [15]. Moreover, we have devised an empirically-based system for the selection of the projection plane that, although not optimal, is able to outperform the method described in [2], which selects the optimal decomposition axis through a computationally expensive trial-and-error approach. As a result of our analysis we show that an adaptive-type entropy encoder might not be an appropriate choice for our proposed method and we expect further gains by replacing the CABAC with a semi-adaptive variant. In our future work we aim to explore efficient ways to transmit the partial frequency table. Furthermore, we aim to define an automated method to identify the set of the most important contexts to be transmitted to the decoder.

Acknowledgments

This research was partially funded by Universitat Autònoma de Barcelona under grant 472-02-1/2017, by the Secretary of Universities and Research (Government of Catalonia) under grant 2017SGR-463, and by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund under grant RTI2018-095287-B-I00 (MINECO/FEDER, UE).

References

[1] D Graziosi, O Nakagami, S Kuma, A Zaghetto, T Suzuki, and A Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020.

[2] Eduardo Peixoto, "Intra-frame compression of point cloud geometry using dyadic decomposition," *IEEE Signal Processing Letters*, vol. 27, pp. 246–250, 2020.

[3] Donald Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[4] Sébastien Lasserre, David Flynn, and Shouxing Qu, "Using neighbouring nodes for the compression of octrees representing the geometry of point clouds," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 145–153.

[5] Diogo C Garcia and Ricardo L de Queiroz, "Intra-frame context-based octree coding for point-cloud geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1807–1811.

[6] Diogo C Garcia, Tiago A Fonseca, Renan U Ferreira, and Ricardo L de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2019.

[7] Ricardo L de Queiroz, Diogo C Garcia, Philip A Chou, and Dinei A Florencio, "Distance-based probability model for octree coding," *IEEE Signal Processing Letters*, vol. 25, no. 6, pp. 739–742, 2018.

[8] Diogo C Garcia and Ricardo L de Queiroz, "Context-based octree coding for point-cloud video," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1412–1416.

[9] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 778–785.

[10] Yingzhan Xu, Wenjie Zhu, Yiling Xu, and Zhu Li, "Dynamic point cloud geometry compression via patch-wise polynomial fitting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2287–2291.

[11] Wenjie Zhu, Yiling Xu, Li Li, and Zhu Li, "Lossless point cloud geometry compression via binary tree partition and intra prediction," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2017, pp. 1–6.

[12] Simone Milani, "Fast point cloud compression via reversible cellular automata block transform," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 4013–4017.

[13] Simone Limuti, Enrico Polo, and Simone Milani, "A transform coding strategy for voxelized dynamic point clouds," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2954–2958.

[14] Rodrigo Rosário and Eduardo Peixoto, "Intra-frame compression of point cloud geometry using boolean decomposition," in *2019 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2019, pp. 1–4.

[15] Eduardo Peixoto, Edil Medeiros, and Evaristo Ramalho, "Silhouette 4d: An inter-frame lossless geometry coder of dynamic voxelized point clouds," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 2691–2695.

[16] C. Loop, Q. Cai, S. Orts Escolano, and P.A. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012, Geneva*, May 2016.

[17] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies - a voxelized point cloud dataset," *ISO/IEC JTC1/SC29/WG1 input document M74006 and ISO/IEC JTC1/SC29/WG11 input document m40059, Geneva*, January 2017.

[18] 3DG, *G-PCC codec description v4*, Tech. Rep., ISO/IEC JTC 1/SC 29/WG 11 input document w18673, 2019.

Chapter 5

Fast run-length compression of point cloud geometry

JCR impact factor: 10.856

Fast Run-Length Compression of Point Cloud Geometry

Dion E. O. Tzamarias, Kevin Chow, Ian Blanes, and Joan Serra-Sagristà, *Senior Member, IEEE*

***Abstract*—The increase in popularity of point-cloud-oriented applications has triggered the development of specialized compression algorithms. In this paper, a novel algorithm is developed for the lossless geometry compression of voxelized point clouds following an intra-frame design. The encoded voxels are arranged into runs and are encoded through a single-pass application directly on the voxel domain. This is done without representing the point cloud via an octree nor rendering the voxel space through an occupancy matrix, therefore decreasing the memory requirements of the method. Each run is compressed using a context-adaptive arithmetic encoder yielding state-of-the-art compression results, with gains of up to 15% over *TMC13*, MPEG's standard for point cloud geometry compression. Several proposed contributions accelerate the calculations of each run's probability limits prior to arithmetic encoding. As a result, the encoder attains a low computational complexity described by a linear relation to the number of occupied voxels leading to an average speedup of 1.8 over *TMC13* in encoding speeds. Various experiments are conducted assessing the proposed algorithm's state-of-the-art performance in terms of compression ratio and encoding speeds.**

***Index Terms*—Point cloud geometry compression, run-length encoding, context-adaptive encoder.**

I. Introduction

Recent advances in 3D acquisition technologies favor point clouds over polygonal meshes for the digital representation of three-dimensional objects. Consisting of a set of attributes and a dense collection of points, placed at precise 3D coordinates, point clouds achieve, in a simple structure, the realistic portrayal of static or even dynamic time-varying scenes. Attesting to the advantages of such data are the numerous point cloud applications in areas such as Augmented/Virtual Reality (AR/VR), self-driving cars, or heritage preservation, to name but a few [1]–[5]. The importance of these applications coupled with the large size and irregular nature of point cloud data have sparked the emergence of an accrescent family of new specialized compression algorithms [6]. As the target of such applications are often smaller devices, such as smartphones in the case of AR/VR, there is an increasing necessity for fast, memory-efficient, low-complexity point cloud compression algorithms [7].

The information that a point cloud carries can be split into its *geometry* data, referring to the 3D coordinates of each point, and its *attribute* data, referring to the value that each point carries, such as the color, the normal or its reflectibility [8]. Therefore, depending on the target, point cloud compression schemes are grouped into geometry or attribute algorithms.

A standard preprocessing step, after acquiring the point cloud data, is the quantization of its geometry information into integer coordinates [9]. This process is referred to as voxelization, where any point in the 3D space is represented as a cubic cell, called voxel and placed in a regular 3D cubic grid. A voxel is said to be occupied when it corresponds to a point of the point cloud, in contrast to unoccupied voxels that form the empty space. Although the voxelization of a point cloud reduces somewhat its original geometry information, additional processing is required for meaningful compression.

The lossy compression of point clouds can attain relatively low bit rates at the expense of introducing distortion on the decompressed data, whereas lossless compression schemes preserve flawlessly the original data at the cost of a higher rate. In the case of dynamically time-varying point cloud scenes, inter-frame encoders exploit redundancies among consecutive point cloud frames. These methods typically lead to higher compression ratios (lower bit rates), though their intra-frame counterparts, which exploit redundancies only within each individual frame, often perform at a lower computational complexity.

The taxonomy of existing lossless point cloud geometry compression designs can be organized according to the manner in which a point cloud is represented throughout the encoding process. These particular point cloud representations, apart from laying the foundations of the encoding process, aim to restrict the number of encoded non-occupied voxels.

The most prevalent representation of point clouds among compression schemes follows the octree decomposition, where the cubic voxel space is recursively subdivided into eight identically-sized cubes called octants [10]. At each iteration, only octants that contain at least one occupied voxel within their volume are further subdivided into children octants, whose occupancy status is indicated through a single bit. The resulting tree structure achieves an a priori compression of the point cloud geometry, though the additional use of transforms, contexts and entropy encoders drastically ameliorates an algorithm's performance.

Some intra-frame methods combine octrees with context-based arithmetic or LZW encoders [11], while other transform-based

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes the the core algorithms in peudocode notation, as well as several proofs of equations. Contact dion.tzamarias@gmail.com for further questions about this work.

methods have octets reversibly transformed by a context-adaptive binary 3D Cellular Automata [12]–[14]. For dynamic point cloud scenes, inter-frame-based designs employ operations on octrees of adjacent frames [15], [16], or sort the octree based on previously encoded frames to promote efficient contexts [17]. In certain approaches, such is the large amount of contexts used in the entropy encoder, that even the context frequency tables have been compressed using an octree representation [11], [18], [19].

Aside from the octree decomposition, there are several different point cloud representations such as the recent G-Arrays known for reduced memory requirements and fast lookup speeds [20]. Unlike the TMC13 lossless geometry compression variant of MPEG, which is also based on the octree, the standardization working group has provided a lossy alternative based on projecting the point cloud on 2D surfaces and employing standard 2D video encoders [5], [8], [21], [22]. Most projection-based compression algorithms represent point cloud geometry through a series of 2D projections using 2D occupancy and depth maps such as the lossy [23]–[25] and lossless methods [26].

Other lossless geometry compression designs can operate straight on the voxel domain by directly processing the coordinates of the occupied voxels, e.g. encoding the residual differences between coordinates of occupied voxels [27]. The advantage of voxel-domain-type encoders lies on the circumvention of intricate preprocessing procedures that are dictated by the representation method, such as the construction of an octree or the calculation of various projection maps.

A sub-family of voxel domain compression methods utilizes the occupancy matrix to represent the point cloud structure. The entire cubic voxel space composed by N^3 voxels is modeled as a three-dimensional $N \times N \times N$ sparse boolean matrix whose elements of value 1 correspond to occupied voxels. Each 2D *slice* of the occupancy matrix, perpendicular to the x, y or z axis is represented by a $N \times N$ boolean array referred to as an image.

Using this representation, the authors of [28] compress the occupancy matrix of a point cloud through a boolean decomposition, and later via a dyadic decomposition and 3D contexts [29]. The latter's impressive results were further improved by extending its intra-frame design to an inter-frame variant [30], as well as by refining the context selection process [31], [32]. An amelioration of the intra-frame design was presented in [33] by removing the dyadic decomposition and utilizing the $N \times N$ projection matrix, calculated through a boolean OR operation across all images of the occupancy matrix perpendicular to the x, y or z axis. The binary projection matrix, as illustrated in Fig. 1b, appears as the shade of the point cloud that falls on the plane perpendicular to the chosen x, y or z axis. Transmitting the binary matrix to the decoder allows to limit the number of encoded voxels as one can deduce the absence of occupied voxels at precise 2D coordinates of all images.

Recent proposals deploy Deep Neural Networks for the geometry compression of point cloud data, achieving very high compression ratios. In the case of the lossless designs, authors follow a hybrid representation of the point cloud, processing voxels through an octree as well as rendering sub-volumes of the voxel space through occupancy matrices [34]–[37]. Though it is important to mention that the complexity of these approaches leads to increased encoding times.

(a) Point cloud (b) Projection matrix (c) Signaling rectangle

Fig. 1. Depiction of a point cloud along with different options to limit encoded voxels. Voxels being encoded are colored white in (b) and (c).

Although providing competitive results in terms of rate, these hybrid and occupancy-type methods render multiple sub-volumes or the entirety of the voxel space, which lead to increased memory requirements and large computational costs. These methods' sequential nature of processing one voxel at a time coupled with the unavoidable encoding of a large number of non-occupied voxels drastically decelerates the compression. As an example, in occupancy-type methods such as [33], it is common that only 2.3% of the processed and encoded voxels are occupied. Likewise, hybrid-type methods such as [37] show that the percentage of occupied voxels in the first level of processed sub-volumes is below 5%.

Our proposed lossless intra-frame geometry compression scheme solves the precedent drawbacks of hybrid and occupancy-type methods while maintaining state-of-the-art compression performance. Using only the point coordinates, the discussed method encodes the point cloud directly on the voxel domain without rendering the voxel space, contributing to its memory-efficient design. Furthermore, through grouping the encoded voxels in runs, the encoding time is drastically decreased. The introduction of several improvements accelerate key calculations, which lower the encoder's computational complexity to the order of occupied voxels, regardless of the number of encoded non-occupied voxels. Lastly, the employment of 3D contexts and an adaptive arithmetic encoder yields significant compression results.

The remainder of this paper is structured in the following manner. Section II summarizes the proposed method. Section III provides descriptions of the core processes. Experimental results are presented and analyzed in Section IV. Conclusions are discussed in Section V. The provided Supplementary material includes pseudocode algorithms and several proofs of equations.

II. Proposed method

The proposed compression scheme, coined *FRL* –for Fast Run-Length–, operates directly on the voxel domain, in the

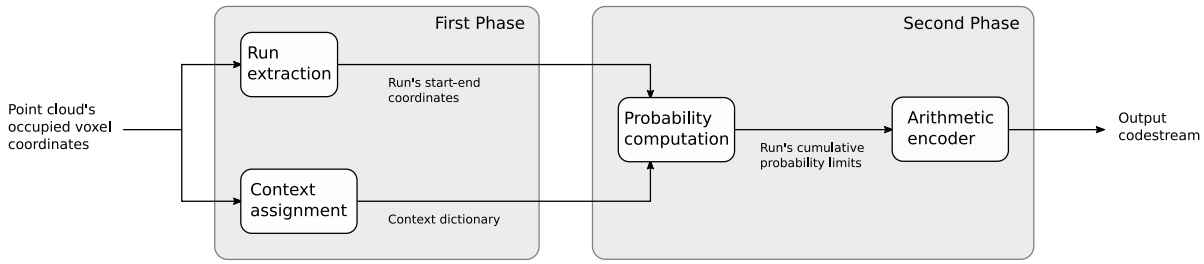


Fig. 2. Simplified version of the proposed single-pass compression scheme indicating the four main processes as well as their inputs and outputs. The processes' names match the title of their dedicated section in the manuscript.

absence of an octree or an occupancy matrix, by encoding runs of voxels through a context-adaptive arithmetic encoder. Its general structure bares similarities to our previous publication [33], which has been improved by virtue of several key novelties. The occupancy matrix is discarded entirely and the compression of individual voxels is replaced in favor of encoding sequences of voxels. Also novel is the acceleration of arduous calculations, required during the encoding process. As a consequence, the memory consumption as well as the overall computational complexity of the algorithm are significantly reduced, yielding a fast, high-performing point cloud geometry compression scheme. The simplified scheme of Fig. 2 indicates the main processes, grouped into two phases, along with each of their inputs and outputs that enable the encoding of sorted point clouds through a single pass. For point clouds that are not already sorted along a desired axis, a sorting preprocessing is required.

The first phase involves the *run-extraction* and *context-assignment* processes. Both receive as input the coordinates of the point cloud's occupied voxels. An essential component that stimulates the accelerated speed of *FRL* consists of forming symbols from runs of voxels rather than individual voxels. This takes place in the process *run-extraction*, which scans the point cloud's geometry information, forming runs of voxels that begin and end within a single 2D plane of the voxel space called a slice. Each run is entirely contained within a single slice and is composed by all non-occupied voxels which are positioned after the end of the previous run and one occupied voxel. I.e., the run ends as soon as the first occupied voxel is found following raster-scan order. The process is concluded after returning the voxel coordinates of each run's starting and ending points.

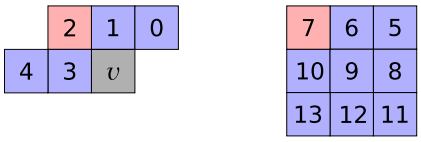
Similar to previous approaches [28], [29], [33], to enhance the prediction accuracy associated to the occupancy of any voxel in a run, the status of its neighboring voxels are also taken into account by means of contexts. The contexts that are employed in this paper are known as 3D contexts, composed by 5 voxels located on the current encoded slice and 9 voxels located on the previous one; cumulating to 14 voxels in total [29]. In Fig. 3, an illustration of the *context map* used in the proposed compression scheme depicts the location of the voxels that form the context, relative to the target voxel whose status is currently examined. Each context is identified according to a unique integer provided by a 14-bit-long bitmask that is defined by the occupancy status of its constituent voxels. An occupied voxel at the position i of the context map contributes 2^i to the context's integer representation. Therefore, 2^{14} unique contexts are deployed in the proposed scheme, whose integer representations range from 0, in the case of the empty context composed solely of non-occupied voxels, to $2^{14}-1$.

On account of contexts playing such an important role in the proposed encoder, the next key process of the phase-one stage involves the assignment of contexts to each voxel. The use of an occupancy matrix coupled with frequent matrix lookups would facilitate the allocation of contexts to each voxel. However, this would impact negatively the overall computational complexity as well as the memory consumption of the algorithm. To our advantage, the proposed *context-assignment* process, in absence of an occupancy matrix, constructs efficiently and inexpensively the context dictionary, a queue data structure that links voxels to their corresponding context. Therefore, the context dictionary accelerates drastically the proposed compression scheme as its size is of the order of occupied voxels and each of its elements is accessed a single time throughout the compression of the point cloud.

The second phase involves the *probability computation* and *arithmetic encoder* processes. Prior to encoding the runs through the *arithmetic encoder* process, each symbol should be associated with an upper and lower cumulative probability limit. This takes place in the *probability computation* process, which utilizes both outputs of the first phase. The introduction of several numerical shortcuts accelerate the calculations of the limits by restricting their complexity to the order of occupied voxels, regardless of the number of non-occupied voxels transmitted. Each run's occurrence probability is calculated through the independent joint probability involving the occupancy status of its component voxels. Furthermore, by means of contexts, the likelihood of a voxel's status is estimated adaptively, using conditional probabilities, through tracking and updating the number of times each specific context occurred to an occupied or a non-occupied voxel.

III. Core processes

The first and second phase processes of the encoder's compression scheme act as the foundations of the proposed method. The functionality, motivation and novelties of each step are



(a) (b)

Fig. 3. Context map employed for encoding voxel v. The 3D contexts are formed by (a) five voxels from the current slice and (b) nine voxels of the previously encoded slice. The 2D coordinates of v and the 9th context coincide. If the non-occupied voxels are noted in blue and the occupied in red, v's context's integer representation is $2^2 + 2^7 = 132$.

detailed in the following subsections. Simplified versions of their related algorithms as well as illustration of the pseudocode notation are provided as supplementary material to demonstrate their functionality.

A. Run extraction

Similar to classical image or text compression, processing groups of data rather than individual symbols is crucial to the speed of an encoder. As it is expected in most hybrid or occupancy-type point cloud geometry compression algorithms, the encoding of non-occupied voxels is unavoidable and often vastly surpass the number of occupied voxels. Therefore, processing and encoding one voxel at a time burdens the computational complexity and encoding speeds of the compression algorithm since they depend fully on the surplus of encoded non-occupied voxels. However, in the hereby introduced method, voxels are not processed individually, but in sequences, accelerating the encoding process and lowering the computational complexity of the algorithm to the order of occupied voxels.

In order to form the runs, the voxel space is first organized into 2D planes, i.e., slices, which are all perpendicular to a predefined axis. Such planes are never rendered as matrices and therefore do not constitute a type of point cloud rendering for the proposed method. For simplicity's sake, throughout the paper, it is assumed that the z-axis has been selected; therefore, the 2D coordinates of any voxel on a slice can be expressed through the x and y axis. Occupied voxels are then grouped by their respective slice such that all of those within a group contain the same z-coordinate. It is with the group's voxel coordinates that runs are formed such that collectively they express the full contents of a slice. Each run is entirely contained within a single slice and is composed primarily by non-occupied voxels, ending on the first encountered occupied voxel, following raster-scan order of the 2D plane. The unique case when a run is permitted to end prematurely on a non-occupied voxel is when it would otherwise extend beyond the end of a slice to reach an occupied voxel. Only in that particular case is the run devoid of an occupied voxel. Thus, given a slice composed of K occupied voxels, the number of runs in said slice is K, unless the slice ends in a non-occupied voxel, in which case it is $K + 1$. Treating such sequences as individual symbols puts in motion the first step of accelerating the encoding process.

It is clear that compression gains can be achieved by reducing the number of non-occupied voxels that are present in the runs. To this end, the encoded area of a slice's 2D voxel space is restricted through the introduction of a signaling rectangle. By registering the x and y coordinates of the north-, south-, west-, and east-most occupied voxels of the point cloud, the occupied voxels are enclosed within a tangential rectangle, indicating the absence of occupied voxels outside the confined area. A realistic example of a point cloud's signaling rectangle is depicted in Fig. 1c. Only the voxels located within the signaling rectangle participate in the runs as all voxels found outside the marked perimeter are dismissed. In the case where no occupied voxels are located within the signaling rectangle, the slice is flagged as empty, and the next plane is processed. Once the encoding process has been completed, the signaling rectangle can be then reproduced in the decoder by transmitting the x and y values of its top, bottom, left and right limits.

An example of the run extraction process is provided in Fig. 4 where each run of the slice is indicated with a different color. All runs are confined within the signaling rectangle, traced with a bold black line. Following raster-scan order, all runs end at occupied voxels, which are represented as squares marked with a dot, except for the last one in blue. The blue run is permitted to end at a non-occupied voxel as it is the last run that is contained within the slice.

The uniform nature of the signaling rectangle makes the computation of the run's length fast and effortless and permits the assignment of contexts without the use of matrices. Yet, the use of the signaling rectangle leads to the processing and encoding of additional non-occupied voxels, compared to the projection matrix; nevertheless, as justified later on and demonstrated in section IV, this does not affect the compression performance of the encoder nor its encoding speed. Since the proposed method encodes the point cloud using directly the coordinates of the occupied voxels, the compression scheme is clearly a voxel-domain-type encoder.

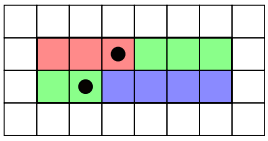


Fig. 4. Example of runs in a slice. Squares correspond to voxels, with the ones marked with a dot being occupied. The signaling rectangle is traced with a bold line. Each run is marked with a different color.

B. Context assignment

The identification of each voxel's context can significantly burden the overall compression scheme in terms of computational complexity. Competitive hybrid-type schemes and occupancy-array-type methods often identify the context of each voxel by rendering multiple volumes or the entire voxel-space through a large matrix such as an occupancy array. The usage of such a matrix burdens the algorithm's memory consumption and impinges on its execution speed. To obtain the context of each

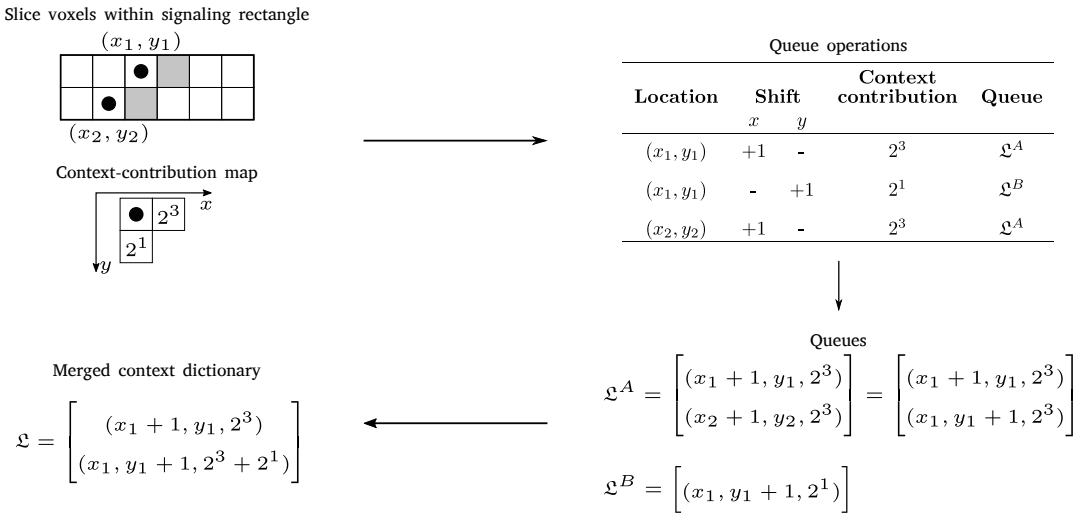


Fig. 5. Example demonstrating the calculation of the context dictionary. Dotted squares correspond to occupied voxels, specified by their coordinates, and shaded squares correspond to voxels linked to a non-empty context. This simplified example considers only the context map positions 1 and 3.

of the W encoded voxels, the status of several elements of the occupancy matrix is assessed, resulting in a complexity of $O(W)$. Given that, usually, the number of occupied voxels V is much smaller than the number of encoded voxels, the use of the occupancy matrix heavily strains the encoding speed and increases the overall computational complexity. In the proposed compression scheme, the complexity of this step is drastically decreased by discarding the occupancy matrix and hence alleviating the memory consumption.

To bypass the aforementioned issues of the context-assignment process, the devised method retrieves the necessary context information through, exclusively, a single-pass over the occupied voxels of a point cloud. This way, a slice's *context dictionary* is defined. Its role is to link the voxels included in any run of the slice with their non-empty context, having the particularity that voxels that are related to the empty context are entirely omitted from the dictionary. Algorithmically, the context dictionary is represented as a queue data structure, $\mathfrak{L}$, composed by a series of triplets that include the two-dimensional coordinates of the voxel as well as the integer representation of its corresponding, non-empty, context. Following the example from the previous subsection and using a simplified set of contexts, Fig. 5 illustrates an instance of a context dictionary, in a slice composed by two occupied voxels (marked with a dot), as well as only two voxels with non-empty contexts (marked in gray). Therefore, if a voxel corresponds to a (non-empty) context, both the coordinates of the voxel, as well as the integer identifier of its context can be accessed. Voxels that are not present in the context dictionary are deduced to be linked to the empty context or outside the boundaries of the signaling rectangle.

To construct a slice's context dictionary without rendering the voxel space, only the 2D coordinates of occupied voxels are used. The computation of $\mathfrak{L}$ is based on the principle that only context map positions related to occupied voxels are required to deduce the integer representation of a voxel's context. For example, referencing Fig. 3, it is only the context map positions of the occupied voxels, i.e., 2 and 7, that are required for the calculation of the integer representation of v's context as $2^2 + 2^7 = 132$. Therefore, a voxel's context can be computed as a sum of the independent contributions of the occupied voxels in its context map.

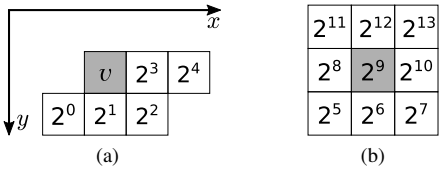


Fig. 6. Context-contribution map: Contribution of the occupied voxel v to the integer representation of the contexts of its surrounding voxels depending on their relative position. (a) Contribution to contexts of voxels located at the same slice as v range from 1 to 2^4, while (b) the ones of voxels located at the next slice of v range from 2^5 to 2^{13}. The darker colored voxels, although located on different slices, share identical (x, y) coordinates.

To uphold the aforementioned single-pass characteristic, it should be possible from the coordinates of an occupied voxel v to retrieve the coordinates of all the voxels that contain v in any position of their context map. Furthermore, each of these voxel coordinates should be linked to the exact contribution of v to their respective contexts. This is achieved via the *context-contribution map* of Fig. 6. By appropriately shifting the coordinates of an occupied voxel, one can match its contribution with the coordinates of the affected voxel. For example, considering an occupied voxel with coordinates (x, y, z), using the context-contribution map it is observed that it contributes 2^2 to the context of the voxel with coordinates $(x+1, y+1, z)$, or that it contributes 2^4 to the context of the voxel with coordinates $(x+2, y, z)$. Regardless of an occupied voxel's global position, each unique shift on its coordinates is therefore linked to a unique contribution. Additionally, the shifted coordinates correspond to those of a voxel whose context has been impacted by said contribution.

Thus, using only the coordinates of occupied voxels, an

individual queue is constructed for each type of shift. By applying a given shift on all occupied voxels within a slice, a queue groups all the coordinates of voxels that have received a unique and identical context-contribution, storing them as triplets. Two examples of such queues are illustrated in Fig. 5 containing in each triplet the aforementioned 2D voxel coordinates and a unique context contribution. In the end, 14 such queues are utilized for the construction of the slice's context dictionary, one for each shift, i.e., one for each unique context contribution.

To illustrate the computation of such queues, the schematic of Fig. 5 provides a visual explanation. For simplicity's sake, only the context map positions 1 and 3 are taken into account, resulting in the simplified context-contribution map indicated in the schematic. As a result, the context dictionary will be calculated using only two queues, $\mathfrak{L}^A$ and $\mathfrak{L}^B$, one for each type of shift as indicated in Fig. 5. Therefore, by shifting the x-coordinate of all occupied voxels by $+1$, it can be seen that the queue $\mathfrak{L}^A$ groups all the coordinates that have an occupied voxel at position 3 of their context-map, therefore receiving a context contribution of 2^3.

By design, each such queue is sorted in ascending order, with the intent to expedite their merging into the context dictionary, composed by triplets with unique coordinates. As illustrated in Fig. 5, the elements of the queues with the same two-dimensional coordinates are integrated into a single triplet whose context integer representation is calculated as the sum of context-contributions of the merged triplets.

The use of the signaling rectangle facilitates the action of determining if a voxel should be added into the context dictionary, devoid of any matrix lookup, rendering the process fast and inexpensive. This would not be possible with a projection matrix since the voxel selection process would require a complexity-raising matrix search to determine if the considered voxel is to be encoded.

Considering that the context dictionary is constructed upon iterating only over occupied voxels, the context identification procedure has a computational complexity of the order of occupied voxels, $O(\mathrm{V})$. This is important as the majority of encoded voxels are found in long runs, of lengths above 80, and most of them in each run are related to the empty context whose frequency is independent from the number of occupied voxels (see Table I). By discarding the use of the occupancy matrix, we significantly decrease the memory usage of our algorithm. The computational complexity and the run time are drastically reduced thanks to the use of context dictionaries.

TABLE I
ON THE LEFT, THE PERCENTAGE OF ENCODED VOXELS FOUND IN RUNS OF LENGTH HIGHER THAN 80 VOXELS. ON THE RIGHT, THE PERCENTAGE OF VOXELS OF EMPTY CONTEXT IN RUNS OF LENGTH ABOVE 80. RESULTS CORRESPOND TO AVERAGES ON ALL FRAMES OF EACH SEQUENCE.

Sequence	**Encoded voxels included in runs of length $\geq$ 80 voxels (%)**	**Voxels linked to empty contexts in runs $\geq$ 80 voxels (%)**
andrew9	97.81	97.25
david9	99.09	98.00
phil9	97.72	97.01
ricardo9	98.31	96.52
sarah9	97.70	96.94
Average	98.13	97.14
longdress	94.44	96.75
loot	96.20	97.24
redandblack	95.21	96.81
soldier	96.92	97.21
Average	95.69	97.00
basketballplayer	98.59	99.35
dancer	98.95	99.53
Average	98.77	99.44
Egyptianmask	99.98	99.99
Shiva	99.97	99.95
Average	99.98	99.97

C. Probability computation

Ensuing the extraction of the runs and the calculation of the context dictionary, each run is compressed via an arithmetic encoder, which first estimates the upper and lower cumulative probability bounds for each run from the context dictionary, and subsequently encodes it. For the purpose of modeling run probabilities, the proposed method takes into account the length of the sequence as well as the contexts of all its composing voxels as illustrated in the Markov chain of Fig. 7.

The Markov chain is expressed by the tree of $2n$ states shown in Fig. 7, where n is the length of the longest-possible run (i.e., the number of remaining uncoded voxels in a slice). The probability of a run is obtained by traversing the tree according to the occupancy of its voxels as follows. Starting at its root, whenever a voxel is occupied, the left child shall be taken; otherwise, the right child shall be taken. The transition probability leading to the ith state is given by the conditional probability of the voxel being occupied given its context c_i.

Each run is composed by a sequence of non-occupied voxels that ends at the first encountered occupied voxel, except for the special case of the last run of a slice. Therefore, the probability of any run is expressed by the independent joint probability of all states that are being traversed starting from the root and ending at any childless node of the tree. Hence, the probability of a run whose length is l and terminates at an occupied voxel is expressed by

$$P(l) = p(1|c_l) \cdot \prod_{i=1}^{l-1} p(0|c_i). \tag{1}$$

In order to arithmetically encode such a sequence of l voxels, one requires the upper and lower cumulative probability limits

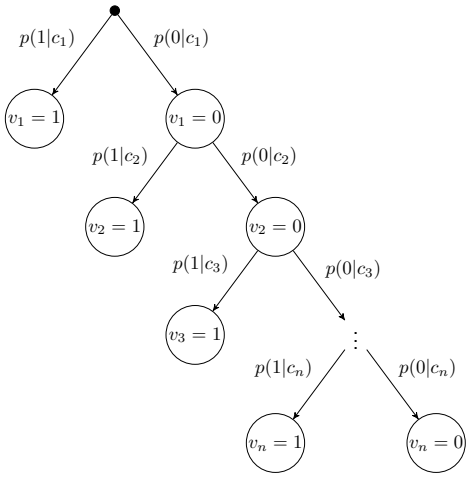


Fig. 7. Markov chain tree, modeling the states of voxels affiliated with runs of lengths shorter or equal to n. Occupied or empty voxels acquire the value of 1 or 0 respectively. The state probability of a given voxel i depends on its context c_i.

of the sequence:

$$\begin{aligned} \text{Low}(l) &= \sum_{r=1}^{l-1} P(r) = \sum_{r=1}^{l-1} p(1|c_r) \cdot \prod_{i=1}^{r} p(0|c_i), \\ \text{High}(l) &= \sum_{r=1}^{l} P(r) = \sum_{r=1}^{l} p(1|c_r) \cdot \prod_{i=1}^{r} p(0|c_i). \end{aligned} \tag{2}$$

At first glance, the calculation of the run's upper and lower cumulative probability bounds appears long and arduous. Therefore, the encoder presented in this paper offers two alternative ways of resolving this problem, each one described in the following subsections.

D. Fast Calculation of Probability Limits

While a direct computation of Eq. 2 is of notable complexity, it can be reformulated into the somewhat simpler Eq. 3 (see Appendix A in the Supplementary Material for the mathematical derivation).

$$\begin{aligned} \text{Low}(l) &= 1 - \prod_{i=1}^{l-1} p(0|c_i), \\ \text{High}(l) &= 1 - \prod_{i=1}^{l} p(0|c_i). \end{aligned} \tag{3}$$

However, by exploiting the adaptive nature of the encoder as described below, the complexity of Eq. 3 can be further decreased to the order of occupied voxels. Instead of examining each voxel in a run, this is achieved by only considering relevant run locations, while maintaining various counts that still allow to obtain identical probability limits.

At each run location linked to a non-empty context c, the counts of non-occupied and occupied voxels linked to any voxel are stored in, respectively, lists $C^{\square}$ and $C^{\blacksquare}$, such that

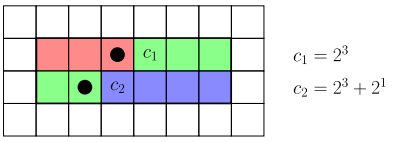


Fig. 8. Example for demonstrating the calculation of the cumulative probability limits of each run. Dotted squares correspond to occupied voxels and voxels with non-empty contexts are marked with their context integer representation, calculated at the example of the *Context assignment* section. Each run is indicated with a separate color.

$p(0|c) = \frac{C_c^{\square}}{C_c^{\blacksquare}+C_c^{\square}}$. I.e., for a non-occupied voxel with context c, the probability $p(0|c)$ is updated by increasing $C_c^{\square}$ by one; for an occupied voxel with context c, the probability $p(0|c)$ is updated by increasing $C_c^{\blacksquare}$ by one.

Conversely, the frequency of non-occupied voxels linked to the empty context is tracked by a single variable $\mathcal{C}$, which is updated at each occurrence of a non-occupied voxel associated to the empty context; i.e., the variable $\mathcal{C}$ is incremented by one. On the contrary, variable $\mathcal{C}$ is divided by two whenever the empty context is linked to an occupied voxel. The count of occupied voxels related to the empty context is assumed to be always equal to 1.

Employing this approach, the calculation of the lower cumulative probability limit for a run composed by k voxels of non-empty context and m voxels linked to the empty context, is given by (see proof in Appendix B in the Supplementary Material):

$$\text{Low}(l) = 1 - \frac{\mathcal{C}}{\mathcal{C}+m} \cdot \prod_{i=1}^{k} p(0|c_i). \tag{4}$$

For example, by applying Eq. 4 one can compute the probability limits of the colored runs marked in Fig. 8, where dotted squares correspond to occupied voxels, and voxels with non-empty contexts are marked with their integer context representation. Therefore, the cumulative probability limits of the red, green and blue run are calculated as $\left[1 - \frac{\mathcal{C}}{\mathcal{C}+2}, 1 - \frac{\mathcal{C}}{\mathcal{C}+3}\right)$, $\left[1 - \frac{\mathcal{C}}{\mathcal{C}+3} \cdot p(0|c_1), 1 - \frac{\mathcal{C}}{\mathcal{C}+4} \cdot p(0|c_1)\right)$ and $\left[1 - \frac{\mathcal{C}}{\mathcal{C}+3} \cdot p(0|c_2), 1\right)$ respectively. It should be highlighted that the upper cumulative limit of the blue run can be deduced to be equal to 1 as it ends on a non-occupied voxel, hence being represented by the final state of the Markov chain tree.

Meanwhile, the employment of the unique manner of tracking the frequencies related to the empty context does not influence the estimation of its conditional probabilities. On the occurrence of an occupied voxel linked to $c = 0$, by forcing its count N_1 to be equal to 1 and dividing $\mathcal{C}$ by 2, the probability $p(1|c = 0)$ can still be computed accurately since

$$p(1|c=0) = \frac{N_1}{N_1 + \mathcal{C}} = \frac{1}{1 + \mathcal{C}/2} = \frac{2}{2 + \mathcal{C}}.$$

This method of encoding the runs is coined fast-mode encoder as opposed to the slower single-mode encoder that is described in the following section.

E. Single-Mode Encoder

During the encoding of runs, the limited numerical precision of processors can cause small numerical errors in the calculations of the probability limits. Most of these numerical errors are harmless to the encoding process as they are recreated identically at the decoder. In some cases though, the difference between the upper and lower probability limits of long runs becomes so small that the arithmetic encoder interprets them as the same value. If left unchecked, this in turn would propagate underflow errors and render the decoder unable to reproduce the original point cloud. Therefore, these underflow cases should be correctly identified and an alternative to the fast-mode encoder should be devised to correctly encode such runs.

The identification of a problematic run simply involves the computation of its probability limits using Eq. 4 and comparing their difference against a specific threshold Q. Only in the cases where the difference becomes lower than the threshold will the fast-mode encoder trigger underflow errors and therefore an alternative type of encoder should be employed.

As the component voxels of a problematic run are unable to be encoded as a single sequence, a process should be carefully devised to rearrange them into admissible runs. One such strategy requires meticulously splitting the run into shorter sub-runs such that their probability limits are appropriately spaced. This procedure is coined single-mode encoding.

The mechanism of splitting a problematic run into sub-runs is performed by having the algorithm iterate over the voxels of the original run, in the order of their appearance in the sequence. At each iteration, a voxel is added to the current sub-run only if the difference of its updated probability limits is larger than the threshold Q. Obviously, the fast calculation of Eq. 4 is abandoned in favor of Eq. 3, which is updated at each inclusion of a new voxel into the current sub-run. In case the addition of a voxel would trigger an underflow, it is added in a new empty sub-run, whereas the old sub-run is terminated at the previous voxel. The following voxels are iteratively added to the new sub-run unless a potential underflow error triggers the initialization of a new sub-run.

The complexity of the single-mode encoder is of the order of the number of encoded voxels $O(W)$, significantly higher than the complexity of the fast-mode encoder. Nevertheless, the single-mode encoder is triggered in the rarest of cases since, on average, it is being employed 0.55 times per point cloud.

IV. Experimental results

To examine the characteristics and behavior of the proposed *FRL* method, several experiments are conducted including a performance comparison against other state-of-the-art geometry compression schemes. The experimentation and analysis are performed on several dynamic and static scenes. The dynamic scenes are provided by the Microsoft upper body [38] and the 8i Labs full body [39] datasets, containing voxelized point cloud sequences placed in a cubic space of dimensions of $512 \times 512 \times 512$ or $1024 \times 1024 \times 1024$ voxels, respectively known as 9 and 10 bit-depth resolution. Two-dimensional depictions of individual point cloud frames of each sequence are provided in Figs. 9 and 10. Each sequence's number of frames, average number of occupied voxels and size of voxel space is given in Table II.

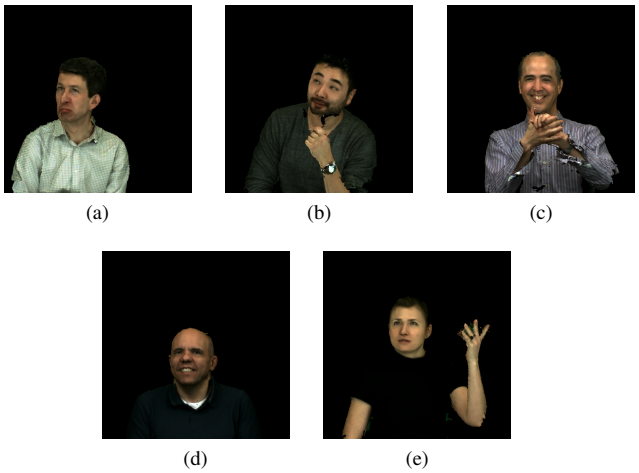


Fig. 9. 2D depiction of single frames from the Microsoft upper body database from the sequences (a) Andrew, (b) David, (c) Phil, (d) Ricardo and (e) Sarah.

The static voxelized scenes are provided by MPEG's test datasets; consisting of the larger point clouds basketball_player_vox11_00000200 and dancer_vox11_00000001 of the Owlii dataset [40], as well as the sparse Egyptian_mask_vox12 and Shiva_00035_vox12 of CTC [41]. These voxelized point clouds are contained in a $2048 \times 2048 \times 2048$ or $4096 \times 4096 \times 4096$ cubic voxel space, respectively known as 11 and 12 bit-depth resolution.

Although the proposed method follows a voxel-domain-based approach, the compression scheme bares many similarities to recent occupancy-array-type algorithms. Therefore, the following high performing –in terms of rate– lossless geometry compression algorithms are included in the comparisons: the intra-frame [33] and [29] methods, as well as inter-frame [32] method, noted as *SP*, *S3D* and *S4D* respectively. The proposed *FRL* method is also compared against competitive lossless hybrid-type methods such as [35] (noted as *MSVDNN*) along with both *VDNN* approaches with and without context extension, respectively [37] and [36]. The *TMC13* v14 MPEG variant [42], which is a recent standard for lossless geometry compression, is also considered in the comparison.

A. Rate Comparison

The comparison, in terms of data compression performance, against state-of-the-art approaches is carried out using all frames of the dynamic scenes as well as the single-frame

(a) Longdress (b) Loot

(c) RedAndBlack (d) Soldier

Fig. 10. 2D depiction of single frames from sequences of the 8i Labs full body database.

TABLE II
POINT CLOUD INFORMATION PER SEQUENCE.

Sequence	Frame count	Average occupied voxels	Size of voxel space
Dynamic scenes			
Microsoft upper body [38]			
andrew9	318	$2.84 \cdot 10^5$	$512 \times 512 \times 512$
david9	216	$3.49 \cdot 10^5$	$512 \times 512 \times 512$
phil9	245	$3.34 \cdot 10^5$	$512 \times 512 \times 512$
phil10	245	$1.49 \cdot 10^6$	$1024 \times 1024 \times 1024$
ricardo9	216	$2.26 \cdot 10^5$	$512 \times 512 \times 512$
ricardo10	216	$1.00 \cdot 10^6$	$1024 \times 1024 \times 1024$
sarah9	207	$2.60 \cdot 10^5$	$512 \times 512 \times 512$
8i Labs full body [39]			
longdress	300	$8.34 \cdot 10^5$	$1024 \times 1024 \times 1024$
loot	300	$7.94 \cdot 10^5$	$1024 \times 1024 \times 1024$
redandblack	300	$7.27 \cdot 10^5$	$1024 \times 1024 \times 1024$
soldier	300	$1.07 \cdot 10^6$	$1024 \times 1024 \times 1024$
Static scenes			
Owlii data [40]			
baskerballplayer	1	$2.93 \cdot 10^6$	$2048 \times 2048 \times 2048$
dancer	1	$2.59 \cdot 10^6$	$2048 \times 2048 \times 2048$
CTC data [41]			
Egyptianmask	1	$2.73 \cdot 10^5$	$4096 \times 4096 \times 4096$
Shiva	1	$1.01 \cdot 10^6$	$4096 \times 4096 \times 4096$

static scenes listed in Table II. The reported results, which are measured in bits per occupied voxel (bpov) and indicated in Table III, are averaged over all the frames of each point cloud sequence.

Before delving into the analysis of the results, it should be noted that the choice of the axis in *FRL*, which determines the order in which the occupied voxels are being processed, is selected through an empirically derived rule. Unless the variance of the points' coordinates along the y-axis is sufficiently larger than along the z-axis, the former axis is always selected. Therefore, the z-axis has been selected for all the sequences of the Microsoft upper body dataset, whereas the y-axis is selected for the rest of the sequences.

Regarding the dynamic scenes, from Table III it is visible that the proposed compression scheme yields very low bit rates, achieving the best reported results for 5 out of the 11 sequences. The herein described encoder significantly outperforms the *TMC13* standard, reporting gains of up to 15%. Additionally, *FRL* outperforms the two recent intra-frame occupancy array-type methods: although the proposed method surpasses *S3D* by a considerable margin, reporting average gains of 9.8%, that margin narrows when focusing on the comparison against *SP*. As mentioned later in section IV-B, *FRL* and *SP* methods follow a similar order of processing the point cloud and use identical 3D contexts, therefore achieving similar results. It appears though, that the usage of the signaling rectangle positively impacts the compression performance of *FRL*, leading to a consistent gain over *SP*. The competitive performance of *FRL* is also highlighted by its comparison against the inter-frame encoder *S4D*: *S4D* is more competitive on the upper body sequences while *FRL* performs better on the full body sequences. Be that as it may, the much larger computational complexity of *S4D*, aggravated by its context selection procedure, renders *FRL* the preferred choice.

Shifting the focus to the neural network hybrid algorithms, the updated version of *VDNN* [37], employing context extension, appears as the most competitive hybrid approach. The authors report compression rates, for a single frame per sequence, of 0.73 and 0.62 bpov on the 10 bit-depth resolution sequences of the upper and full body datasets respectively [38], [39]. The results of the neural network methods [36] and [35], presented in Table III, are directly taken from the cited papers, however the authors do not specify the number of frames that were tested in each sequence. From Table III it is clear that *FRL* is outperformed by *VDNN* for the sequences used for its evaluation in [36]. Though, this competitive performance is penalized by its lengthy run time, provided in [35], which is many orders of magnitude larger than *FRL*'s, as indicated later in Section IV-C. On the other hand, *FRL* outperforms *MSVDNN* on sequences of the Microsoft upper body dataset while the latter is more competitive only on sequences of the 8i Labs' full body dataset.

Although the previously mentioned neural network approach is an accelerated version of *VDNN*, *MSVDNN* is several orders of magnitude more computationally expensive than the proposed approach. This is highlighted by the encoding speed measurements of *MSVDNN* when implemented on the optimized PyTorch software and executed on a high-performance GeForce RTX 2080 GPU [35].

In the case of larger, dense static scenes, such as basketball or dancer, the performance of FRL does not appear to be hindered. On these point clouds it is observed that the average

TABLE III
RATE COMPARISON BETWEEN PROPOSED METHOD *FRL* AND STATE OF THE ART. MEASUREMENTS ARE AVERAGED OVER ALL AS WELL AS THE 100 FIRST FRAMES OF EACH SEQUENCE AND DISPLAYED IN BITS PER OCCUPIED VOXEL [BPOV]

Sequence	All frames (Intra coders)				Unspecified number of frames (Intra coders)		First 100 frames (Intra coder)	First 100 frames (Inter coder)
	FRL	**SP [33]**	**S3D [29]**	**TMC13 [42]**	**VDNN [36]**	**MSVDNN [35]**	**FRL**	**S4D [32]**
andrew9	**1.00**	1.02	1.12	1.13	-	-	1.01	**0.95**
david9	**0.94**	0.96	1.05	1.07	-	-	**0.94**	**0.94**
phil9	1.02	1.04	1.14	1.17	**0.92**	-	**1.02**	**1.02**
phil10	0.95	-	-	-	**0.83**	1.02	0.95	-
ricardo9	0.93	0.95	1.03	1.07	**0.72**	-	0.94	**0.90**
ricardo10	0.89	-	-	-	**0.75**	0.95	0.90	-
sarah9	**0.95**	0.97	1.06	1.07	-	-	0.96	**0.92**
Average	0.95	0.99	1.08	1.10	**0.81**	0.99	0.96	**0.95**
longdress	**0.86**	0.89	0.95	1.02	-	-	**0.86**	0.88
loot	0.83	0.86	0.92	0.97	0.64	**0.63**	**0.82**	0.84
redandblack	0.94	0.96	1.03	1.09	**0.73**	0.87	**0.93**	0.94
soldier	**0.88**	0.91	0.97	1.04	-	-	0.88	**0.65**
Average	0.88	0.91	0.97	1.03	**0.69**	0.75	0.87	**0.83**
basketballplayer	**0.80**	-	-	0.90	-	-	-	-
dancer	**0.77**	-	-	0.89	-	-	-	-
Average	**0.79**	-	-	0.90	-	-	-	-
Egyptianmask	18.20	-	-	**11.78**	-	-	-	-
Shiva	15.11	-	-	**9.68**	-	-	-	-
Average	16.66	-	-	**10.73**	-	-	-	-

compression gains over TMC13 are at 12%. However, in sparse point clouds such as Egyptianmask and Shiva, FRL is in deficit in terms of compression performance. This might very well be due to the limited sub-volume of the voxel space that is covered by the employed contexts, since they are composed only by 14 voxels. That is to say, in sparse point clouds such is the isolation of individual voxels that the contexts struggle to extract meaningful information from local voxel neighborhoods. Such a phenomenon has been studied in the hybrid geometry encoder [37], where a decrease in performance over sparse data was also reported. To somewhat alleviate such a drawback, the authors proposed a context extension strategy which increases the volumetric space covered by the contexts.

B. Projection matrix vs signaling rectangle

The similarity in terms of rate between the proposed *FRL* and *SP*, whose performance are compared in section IV-A, is due to the use of identical contexts. The main differences between these two approaches that could affect their performance in terms of rate is the use of different parameters in the arithmetic encoder and the replacement of the projection matrix with the signaling rectangle in *FRL*. Therefore, the demonstrated gains over *SP* are attributed to both the signaling rectangle and an optimized arithmetic encoding. That said, if the two methods both employed a signaling rectangle, and identical encoder parameters, the final upper and lower probability limits determined by their arithmetic encoder related to the entire point cloud would be identical.

In the results of Table IV it is noticeable that the encoding of the signaling rectangle is much more cost-efficient than the encoding of the projection matrix. This is because the corners of the signaling rectangle are transmitted to the decoder using 16 bits apiece. On the other hand in [33], the 2D projection matrix, which resembles a binary image with intricate curves and forms, is much more costly to encode.

However, the advantage of the aforementioned matrix is linked to its ability to reduce the number of encoded voxels significantly more than the signaling rectangle. In the last column of Table IV it is evident that the proposed method, through the signaling rectangle, encodes at least 76% more voxels than in [33] using the projection matrix. That said, almost the entirety of the additionally encoded non-occupied voxels are associated with the empty context, which predicts extremely accurately the occupancy status of a voxel. Therefore, the low coding cost of the signaling rectangle counteracts the small increase in rate attributed to the surplus of encoded voxels. Hence, by introducing the signaling rectangle in the proposed compression scheme, a significant reduction in memory consumption, computational complexity and encoding speed can be achieved without compromising its rate-related performance. Due to such drawbacks in memory consumption, the *SP* implementation (in Matlab) is unable to process very large point clouds, which is why the results for the static scenes are absent from Table IV.

C. Run-Time Comparison

The proposed point cloud encoder is designed with the aim to significantly reduce encoding times and computational complexity while maintaining a very competitive geometry compression performance. In the following experiment, the

TABLE IV
CODING COST OF PROJECTION MATRIX VERSUS SIGNALING RECTANGLE. THE RIGHT-MOST COLUMN INFORMS ON THE PERCENTAGE SURPLUS OF ENCODED VOXELS IF THE SIGNALING RECTANGLE IS USED INSTEAD OF THE PROJECTION MATRIX. MEASUREMENTS ARE AVERAGED OVER ALL FRAMES OF EACH SEQUENCE.

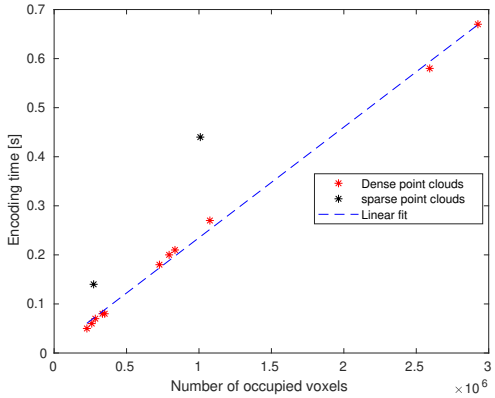
Sequence	Projection matrix (bpov)	Signaling rectangle (bpov)	Surplus of encoded voxels (%)
andrew9	0.0170	$22.5 \cdot 10^{-5}$	145.15
david9	0.0188	$18.3 \cdot 10^{-5}$	178.19
phil9	0.0221	$19.2 \cdot 10^{-5}$	80.31
ricardo9	0.0168	$28.3 \cdot 10^{-5}$	84.29
sarah9	0.0186	$24.6 \cdot 10^{-5}$	120.32
Average	0.0187	$22.6 \cdot 10^{-5}$	121.65
longdress	0.0031	$7.7 \cdot 10^{-5}$	75.52
loot	0.0049	$8.1 \cdot 10^{-5}$	89.50
redandblack	0.0060	$8.8 \cdot 10^{-5}$	89.68
soldier	0.0034	$6.0 \cdot 10^{-5}$	92.05
Average	0.0044	$7.7 \cdot 10^{-5}$	86.69

proposed *FRL* is coded in C and the timing performance of each of the algorithms was evaluated on an 8 core 2.8GHz Intel Core i7-7700HQ CPU.

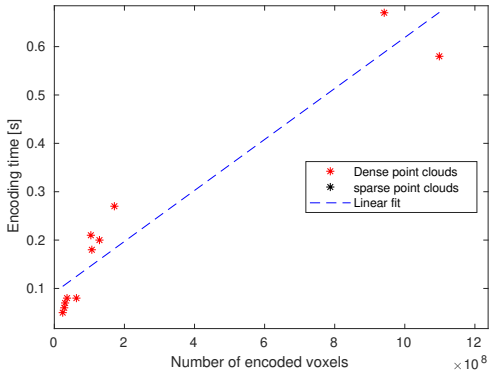
Given that the complexity of the proposed algorithm is of the order of occupied voxels $O(V)$, the increase in the number of encoded non-occupied voxels caused by the use of the signaling rectangle does not affect the overall encoding time. As illustrated in Fig. 11, the encoding time of the proposed algorithm on dense point clouds (red dots), is linearly correlated with the number of occupied voxels and not the number of encoded voxels, confirming the effect of Eq. 4 on the speed of the compression scheme. As reported in Table II, the percentage of occupied voxels is between 0.0004% and 0.26% of the total number of voxels, which illustrate the very competitive behavior of our approach. In the case of sparse data (black dots) the linear correlation of Fig. 11 seems disrupted even though the single mode encoder is not triggered more often than in the case of dense data. This increase in encoding time is due to the negligible terms in the complexity calculation related to the increased number of voxels linked to non-empty contexts (which can be at most 14 times the number of occupied voxels on a slice).

Table V marks the encoding time measurements on static as well as on dynamic scenes, which are provided as averages over all frames of each sequence. The proposed *FRL* method has only been compared against *TMC13* v14 (implemented in C++) since the latter is the most competitive among the algorithms used in the experiments, in terms of execution time. To assure a fair comparison, both encoders have been build using exactly same optimization level: -O3 and -march=native.

In addition to the average encoding time of each sequence, the initial sorting of the point cloud along a selected axis is also taken into account in the case of *FRL*. Following the reported results it is observed that for dense point clouds, such as the dynamic scenes and the Owlii data, *FRL* is on average 1.4



(a) Occupied voxels vs encoding time (average)



(b) Encoded voxels vs encoding time (average). Sparse data points are out of view with an encoding time of sub 0.44s and a number of encoded voxels of $2.9 \cdot 10^{10}$ and $5.6 \cdot 10^{10}$

Fig. 11. Observed relation between encoding time and voxel count.

times faster than *TMC13*. In the case of sparse scenes such as Egyptianmask and Shiba, the increase in encoding speed of the proposed method over *TMC13* is almost three-fold.

In the case of the hybrid-based methods, the authors report that, on average, the encoding speeds of *MSVDNN* as well as both *VDNN* with and without context extension, were respectively, 14, 3186 and 658 times slower than *TMC13* [35]. These averages were computed on a test set of dynamic and static scenes of a bit depth of 10 and 9. Therefore, since our experimental results have demonstrated that, on average, *FRL* is more than 1.3 times faster than *TMC13*, on a similar test set, it is safe to say that the proposed encoder is many orders of magnitude faster than the aforementioned hybrid approaches.

V. CONCLUSION

A novel lossless point cloud geometry compression algorithm is proposed in this paper. Its intra-frame design permits its use on static and dynamic scenes. The proposed encoder is applied directly on the voxel domain of the point cloud, therefore limiting requirements on memory consumption by avoiding the use

TABLE V
ENCODING TIME COMPARISON BETWEEN PROPOSED METHOD *FRL* AND STATE OF THE ART. MEASUREMENTS ARE AVERAGED ON ALL FRAMES OF EACH SEQUENCE AND DISPLAYED IN SECONDS.

Sequence	**FRL**			**TMC13 [42]**	**Speedup over TMC13**
	Sorting	Encoding	Total		
andrew9	0.005	0.07	**0.07**	0.10	1.4
david9	0.006	0.08	**0.09**	0.13	1.4
phil9	0.006	0.08	**0.08**	0.12	1.5
ricardo9	0.004	0.05	**0.06**	0.08	1.3
sarah9	0.005	0.06	**0.06**	0.09	1.5
Average	0.005	0.07	**0.07**	0.10	1.4
longdress	0.017	0.21	**0.22**	0.31	1.4
loot	0.016	0.20	**0.21**	0.29	1.4
redandblack	0.014	0.18	**0.20**	0.28	1.4
soldier	0.021	0.27	**0.29**	0.40	1.4
Average	0.017	0.22	**0.23**	0.32	1.4
basketball	0.061	0.67	**0.74**	1.05	1.4
dancer	0.054	0.58	**0.64**	0.92	1.4
Average	0.058	0.63	**0.69**	0.99	1.4
Egyptianmask	0.009	0.14	**0.15**	0.48	3.2
Shiva	0.025	0.44	**0.47**	1.30	2.8
Average	0.017	0.29	**0.31**	0.89	2.9

of an occupancy matrix or of an octree representation. Instead of encoding one voxel at a time, entire runs of voxels are encoded using a run-length-adaptive arithmetic encoder. The proposal of several novelties that accelerate key calculations, regarding the cumulative probability limits of each run, render the algorithm, on average, 1.8 times faster than *TMC13* and of low computational complexity. The introduction of the signaling rectangle restricts the number of encoded voxels while the employment of 3D contexts yields competitive state-of-the-art performance in terms of bit-rate. A series of experimental results confirm the preceding claims, including compression ratio (or bit-rate) and execution time comparison against state-of-the-art compression schemes. Future research will be steered towards ameliorating the algorithm's performance on sparse data, improving the contexts for each point cloud and optimizing the axis selection choice without impacting heavily the overall complexity of the algorithm.

ACKNOWLEDGMENT

This research was partially funded by Universitat Autònoma de Barcelona under grant 472-02-1/2017, by the Secretary of Universities and Research (Government of Catalonia) under grant 2017SGR-463, and by the Spanish Ministry of Economy and Competitiveness, the Spanish Ministry of Science and Innovation, and the European Regional Development Fund under grants RTI2018-095287-B-I00 (MINECO/FEDER, UE) and PID2021-125258OB-I00 (MICINN/FEDER, UE).

REFERENCES

[1] D. Ni, A. Y. Nee, S.-K. Ong, H. Li, C. Zhu, and A. Song, "Point cloud augmented virtual reality environment with haptic constraints for teleoperation," *Transactions of the Institute of Measurement and Control*, vol. 40, no. 15, pp. 4091–4104, 2018.

[2] Y. Wang, S. Zhang, B. Wan, W. He, and X. Bai, "Point cloud and visual feature-based tracking method for an augmented reality-aided mechanical assembly system," *The International Journal of Advanced Manufacturing Technology*, vol. 99, no. 9, pp. 2341–2352, 2018.

[3] K. Kohira and H. Masuda, "Point-cloud compression for vehicle-based mobile mapping systems using portable network graphics." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. IV-2/W4, pp. 99–106, 2017.

[4] A. Ullah, M. Watanabe, A. Kubo *et al.*, "Analytical point-cloud based geometric modeling for additive manufacturing and its application to cultural heritage preservation," *Applied Sciences*, vol. 8, no. 5, p. 656, 2018.

[5] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020.

[6] C. Cao, M. Preda, V. Zakharchenko, E. S. Jang, and T. Zaharia, "Compression of sparse and dense dynamic point clouds–methods and standards," *Proceedings of the IEEE*, 2021.

[7] F. Qian, B. Han, J. Pair, and V. Gopalakrishnan, "Toward practical volumetric video streaming on commodity smartphones," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, 2019, pp. 135–140.

[8] H. Liu, H. Yuan, Q. Liu, J. Hou, and J. Liu, "A comprehensive study and comparison of core technologies for MPEG 3-d point cloud compression," *IEEE Transactions on Broadcasting*, vol. 66, no. 3, pp. 701–717, 2019.

[9] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3886–3895, 2017.

[10] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[11] D. C. Garcia and R. L. de Queiroz, "Intra-frame context-based octree coding for point-cloud geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1807–1811.

[12] S. Milani, "Fast point cloud compression via reversible cellular automata block transform," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 4013–4017.

[13] S. Limuti, E. Polo, and S. Milani, "A transform coding strategy for voxelized dynamic point clouds," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2954–2958.

[14] S. Milani, E. Polo, and S. Limuti, "A transform coding strategy for dynamic point clouds," *IEEE Transactions on Image Processing*, vol. 29, pp. 8213–8225, 2020.

[15] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 778–785.

[16] R. L. de Queiroz, D. C. Garcia, P. A. Chou, and D. A. Florencio, "Distance-based probability model for octree coding," *IEEE Signal Processing Letters*, vol. 25, no. 6, pp. 739–742, 2018.

[17] D. C. Garcia and R. L. de Queiroz, "Context-based octree coding for point-cloud video," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1412–1416.

[18] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2019.

[19] D. C. Garcia, T. A. Fonseca, and R. L. De Queiroz, "Example-based super-resolution for point-cloud video," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2959–2963.

[20] H. Roodaki, M. Dehyadegari, and M. N. Bojnordi, "G-arrays: Geometric arrays for efficient point cloud processing," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 1925–1929.

[21] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[22] E. S. Jang, M. Preda, K. Mammou, A. M. Tourapis, J. Kim, D. B. Graziosi, S. Rhyu, and M. Budagavi, "Video-based point-cloud-compression standard in MPEG: from evidence collection to committee draft [standards in a nutshell]," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 118–123, 2019.

[23] I. Tabus, E. C. Kaya, and S. Schwarz, "Successive refinement of bounding volumes for point cloud coding," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–6.

[24] J. Xiong, H. Gao, M. Wang, H. Li, K. N. Ngan, and W. Lin, "Advanced geometry surface coding for dynamic point cloud compression," *arXiv preprint arXiv:2103.06549*, 2021.

[25] Y. Xu, W. Zhu, Y. Xu, and Z. Li, "Dynamic point cloud geometry compression via patch-wise polynomial fitting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2287–2291.

[26] E. C. Kaya, S. Schwarz, and I. Tabus, "Refining the bounding volumes for lossless compression of voxelized point clouds geometry," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3408–3412.

[27] W. Zhu, Y. Xu, L. Li, and Z. Li, "Lossless point cloud geometry compression via binary tree partition and intra prediction," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2017, pp. 1–6.

[28] R. Rosário and E. Peixoto, "Intra-frame compression of point cloud geometry using boolean decomposition," in *2019 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2019, pp. 1–4.

[29] E. Peixoto, "Intra-frame compression of point cloud geometry using dyadic decomposition," *IEEE Signal Processing Letters*, vol. 27, pp. 246–250, 2020.

[30] E. Peixoto, E. Medeiros, and E. Ramalho, "Silhouette 4d: An inter-frame lossless geometry coder of dynamic voxelized point clouds," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 2691–2695.

[31] E. Ramalho and E. Peixoto, "Context selection for lossless compression of bi-level images," in *XXXVIII Simpsio Brasileiro de Telecomunicacoes e Processamento de Sinai*, 2021.

[32] E. Ramalho, E. Peixoto, and J. E. G. de Medeiros, "Silhouette 4d with context selection: Lossless geometry compression of dynamic point clouds," *IEEE Signal Processing Letters*, 2021.

[33] D. E. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagristà, "Compression of point cloud geometry through a single projection," in *2021 Data Compression Conference (DCC)*. IEEE, 2021, pp. 63–72.

[34] E. C. Kaya and I. Tabus, "Neural network modeling of probabilities for coding the octree representation of point clouds," *arXiv preprint arXiv:2106.06482*, 2021.

[35] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Multiscale deep context modeling for lossless point cloud geometry compression," in *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2021, pp. 1–6.

[36] ——, "Learning-based lossless compression of 3d point cloud geometry," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4220–4224.

[37] ——, "Lossless coding of point cloud geometry using a deep generative model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4617–4629, 2021.

[38] C. Loop, Q. Cai, S. O. Escolano, and P. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012, Geneva*, May 2016.

[39] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies - a voxelized point cloud dataset," *ISO/IEC JTC1/SC29/WG1 input document M74006 and ISO/IEC JTC1/SC29/WG11 input document m40059, Geneva*, January 2017.

[40] Y. Xu, Y. Lu, and Z. Wen, "Owlii Dynamic human mesh sequence dataset," ISO/IEC JTC1/SC29/WG11, 120th MPEG Meeting, Macau, Input document m41658, Oct. 2017.

[41] C. Tulvan, A. Gabrielli, and M. Preda, "Datasets update on Point Cloud compression for cultural objects," ISO/IEC JTC1/SC29/WG11, 115th MPEG meeting, Geneva, Input document m38678, May 2016.

[42] "C++ MPEG group TMC13 implementation," [Online] Available: https://github.com/MPEGGroup/MPEG-pcc-tmc13, accessed: May 2022.

Supplementary Material of: Fast Run-Length Compression of Point Cloud Geometry

Dion E. O. Tzamarias, Kevin Chow, Ian Blanes, and Joan Serra-Sagristà, *Senior Member, IEEE*

I. Introduction

This document provides supplementary material for the paper titled *Fast Run-Length Compression of Point Could Geometry*. Specifically, several algorithms are provided in pseudocode notation, detailing the function of the key processes of the proposed scheme, illustrated in Figure 1. Detailed proofs of equations (3) and (4) of the journal paper are also included as an addendum. The organization of the document is as follows: In Section II and III, the phase one and two processes are detailed algorithmically. The document is concluded with two appendix sections detailing the proofs of the aforementioned equations.

II. First-phase processes

The processes *run extraction* and *context assignment* included in the first phase of the encoder's compression scheme are algorithmically displayed in pseudocode notation to demonstrate their functionality. Table I illustrates some examples of the pseudocode nomenclature. All pseudocode variables are represented in mathtext while simple variables are noted in lower case Latin letters, lists in upper case Latin letters and queues in the fraktur font. To access an element of a list a subscript is added, specifying in mathtext the index of the element such as in line 5 of the algorithm. Textmode is usually reserved for global variables or naming purposes such as in the case of simple variables, where the name is placed in subscript, or in lists and queues, where it is positioned in superscript.

TABLE I
Examples of nomenclature used in pseudocode

	Notation	Definition
Variables	p, l	Simple variables.
	P, L	Lists
	$\mathfrak{P}, \mathfrak{L}$	Queues
	$v_{\mathrm{y}}, v_{\mathrm{x}}$	Naming simple variables.
	$P^{\mathrm{low}}, \mathfrak{L}^{\mathrm{up}}$	Naming lists and queues.
	P_k	The kth element of list P
	$C^{\square}, p^{\square}$	Associated to non-occupied voxels.
	$C^{\blacksquare}, n^{\blacksquare}$	Associated to occupied voxels.
Constants	F	Index of first occupied slice.
	E	Index of last occupied slice.
	L	Occupancy left coordinate limit.
	R	Occupancy right coordinate limit.
	T	Occupancy upper coordinate limit.
	B	Occupancy lower coordinate limit.
	Q	Single mode encoding threshold.
Operations	**APPEND**	Add an element at the end of a list.
	PUSH	Place an element at the end of the queue.
	PEEK	Look at the first element of the queue without removing it.
	POP	Remove the first element from the queue.
	or	Logical or.
	and	Logical and.
Miscellaneous	$\leftarrow$	Assign a value to an object.
	$=$	Evaluate if equal.
	$\neq$	Evaluate if not equal.
	//	Comment
	$\#$	Number of elements in list or queue.
	$\mathcal{C}$	Special object.

A. Run extraction

The process of forming runs out of the list of *occupied* voxels of a slice, noted as V, is demonstrated in Algorithm 1. The first run of a slice starts at the upper left corner of the signaling rectangle with coordinates $(\mathrm{T},\ \mathrm{L})$ and ends at the slice's first occupied voxel V_0. Once the run has been encoded, the new run starts from the next voxel after V_0, following a raster scan order. Lines 7-11 ensure that the start of the new run is located within the signaling rectangle whose top, bottom, left and right limits are given by $\mathrm{T},\ \mathrm{B},\ \mathrm{L}$ and R respectively. Meanwhile the new ending point is the next occupied voxel V_1. The process is repeated until the ending point of the last run is the lower right corner of the signaling rectangle with coordinates $(\mathrm{R},\ \mathrm{B})$.

B. Context assignment

Algorithm 2 details the process *context assignment*, which using only the coordinates of occupied voxels, constructs the context dictionary $\mathfrak{L}$. For simplicity sake, the pseudocode designs the context dictionary of a slice that links voxels to the contexts characterized by occupancies in the first two positions of the context map displayed as 0 and 1 in Figure 2a. As a result $\mathfrak{L}$ is formed by two queues noted as $\mathfrak{L}^A$ and $\mathfrak{L}^B$ (linked to the context contributions of 2^0 and 2^1 respectively), that are associated to two unique coordinate shifts on the occupied voxels clarified by the contribution map in Figure 2b.

As an example, in lines seven and ten of Algorithm 2, the queue $\mathfrak{L}^A$ shifts the coordinates of occupied voxels V by $(x-1, y+1)$. This obtains the coordinates of the voxels that contain an occupied voxel at location 0 of their context-

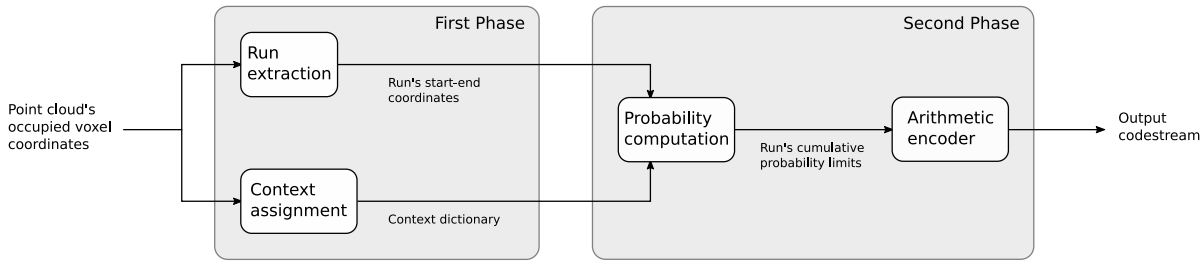


Fig. 1. Simplified version of the proposed single-pass compression scheme indicating the four main processes as well as their inputs and outputs.

map therefore receiving a contribution of 2^0 to their context's integer representation. Each 2D shifted coordinates are stored in $\mathfrak{L}^A$, in raster-scan order, along with the unique contribution in the form of triplets. A series of queue operators, marked in bold textmode are defined in Table I. The queues $\mathfrak{L}^A$ and $\mathfrak{L}^B$ are then merged into the context dictionary such that triplets with the same two dimensional coordinates are integrated into a single triplet whose context integer representation is calculated as the sum of context-contributions of the merged triplets. The signaling rectangle facilitates identification of non-transmittable voxels which are thus omitted from the context-dictionary, via the *if* conditions of lines 6 and 9.

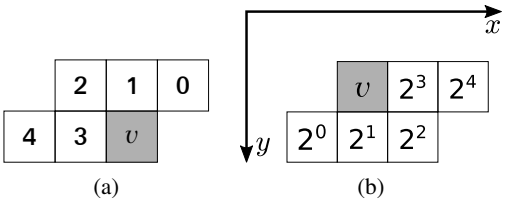


Fig. 2. (a) Context-map used to determine the integer representation of the context of voxel v. (b) Context-contribution-map illustrating the contribution of occupied voxel v to contexts of voxels located at the same slice as v, ranging from 1 to 2^4.

III. Second-phase processes

The probability limits of a run are computed according to the pseudocode of Algorithm 3. As indicated in Table I black ■ or white □ squares are placed in superscript for naming purposes to clarify when the object is associated to occupied or non-occupied voxels respectively. One can note that the product term of the limits associated to non-empty contexts is computed sequentially in the *while* loop of lines 7 to 15. The context-dictionary $\mathfrak{L}$ facilitates the retrieval of a voxel's non-empty context c whose count $C_c^{\square}$ is immediately updated during the calculation of the product. In the other hand though, the term linked to the empty context is non-iteratively calculated in line 18. This is achieved by keeping track of the number of voxels linked to a non-empty context and easily computing the total length of the run through the signaling rectangle in lines 13 and 1 respectively.

In the case a potential underflow error is detected, by comparing the difference of the probability pair against threshold Q,

Algorithm 1 The function **RUN-EXTRACTION** is a run length encoder using an arithmetic encoder.*Input* is a list V composed by tuples $(v_{\text{x}}, v_{\text{y}})$ of the x and y-coordinates of all the occupied voxels located on a single slice. The lists $C^{\square}, C^{\blacksquare}$ and the integer $\mathcal{C}$ allow the calculation of the probability limits associated to a specific run, whereas the list S^{mod} flags the use of the correct auxiliary function to avoid any underflow errors in the arithmetic encoder. *Output* $C^{\square}, C^{\blacksquare}, S^{\text{mod}}, \mathcal{C}$ are updated for later use.

INPUT: $V, C^{\square}, C^{\blacksquare}, S^{\text{mod}}, \mathcal{C}$
OUTPUT: $C^{\square}, C^{\blacksquare}, S^{\text{mod}}, \mathcal{C}$
1: $\mathfrak{L} \leftarrow$ **CONTEXT-ASSIGNMENT**(V)
// Start of first run
2: $(v_{\text{sx}}, v_{\text{sy}}) \leftarrow (\text{L}, \text{T})$
// Initializing end of first run
3: $(v_{\text{ex}}, v_{\text{ey}}) \leftarrow (\text{L}, \text{T})$
// Iterate over occupied voxels of slice
4: **for** i **from** 0 **to** $\#V - 1$ **do**
// End of new run at next occupied voxel
5: $(v_{\text{ex}}, v_{\text{ey}}) \leftarrow V_i$
// encode run starting at $(v_{\text{sx}}, v_{\text{sy}})$ and ending at $(v_{\text{ex}}, v_{\text{ey}})$
6: $C^{\square}, C^{\blacksquare}, \mathcal{C}, S^{\text{mod}} \leftarrow$ **FASTLIMITS**$(v_{\text{sx}}, v_{\text{sy}}, v_{\text{ex}}, v_{\text{ey}}, b = 1, \mathfrak{L}, C^{\square}, C^{\blacksquare}, \mathcal{C}, S^{\text{mod}})$
// Calculate start of new run at next voxel
7: **if** $v_{\text{ex}} + 1 > \text{R}$ **then**
8: $(v_{\text{sx}}, v_{\text{sy}}) \leftarrow (\text{L}, v_{\text{ey}} + 1)$
9: **else**
10: $(v_{\text{sx}}, v_{\text{sy}}) \leftarrow (v_{\text{ex}} + 1, v_{\text{ey}})$
11: **end if**
12: **end for**
13: **if** $v_{\text{ex}} \neq \text{R}$ **or** $v_{\text{ey}} \neq \text{B}$ **then**
14: $(v_{\text{sx}}, v_{\text{sy}}) \leftarrow (v_{\text{ex}}, v_{\text{ey}})$
15: $(v_{\text{ex}}, v_{\text{ey}}) \leftarrow (\text{R}, \text{B})$
// encode run starting at $(v_{\text{sx}}, v_{\text{sy}})$ and ending at $(v_{\text{ex}}, v_{\text{ey}})$
16: $C^{\square}, C^{\blacksquare}, \mathcal{C}, S^{\text{mod}} \leftarrow$ **FASTLIMITS**$(v_{\text{sx}}, v_{\text{sy}}, v_{\text{ex}}, v_{\text{ey}}, b = 0, \mathfrak{L}, C^{\square}, C^{\blacksquare}, \mathcal{C}, S^{\text{mod}})$
17: **end if**

the single mode encoder is triggered in line 49. The single-mode encoder, demonstrated in Algorithm 4, partitions the original run into multiple sub-runs which are individually encoded by an arithmetic encoder without causing any underflow errors. In the course of encoding the point cloud, the partitioned runs are signaled and registered at the header bitstream through the list S^{mod}, as demonstrated in lines 50

to 52 of Algorithm 3. This allows the decoder to adapt its decoding strategy depending on the mode of encoder used on the run.

Algorithm 2 Algorithm **CONTEXT-ASSIGNMENT** obtains the non-empty context of each voxel through the contexts binary mask. *Input* V is a list of the slice's occupied voxel coordinates in raster-scan order. *Output* $\mathfrak{L}$ is a queue of triplets. For a given triplet the first 2 elements are the x and y-coordinates of a voxel with a non-zero context, whereas the 3rd element is the (non-empty) context of that voxel. In this pseudocode we take into account only 2 context contributions for brevity.

INPUT: V
OUTPUT: $\mathfrak{L}$
$\mathfrak{L} \leftarrow$ **Empty Queue**
$\mathfrak{L}^A \leftarrow$ **Empty Queue**
$\mathfrak{L}^B \leftarrow$ **Empty Queue**
for i **from** 0 **to** $\#V - 1$ **do**
  $(v_{\mathrm{x}}, v_{\mathrm{y}}) \leftarrow V_i$
  // Push tuples to sorted queue $\mathfrak{L}^A$
  if $v_{\mathrm{x}} - 1 \geq \mathrm{L}$ **and** $v_{\mathrm{x}} - 1 \leq \mathrm{R}$ **and** $v_{\mathrm{y}} + 1 \geq \mathrm{T}$ **and** $v_{\mathrm{y}} + 1 \leq \mathrm{B}$ **then**
    PUSH triplet $\left(v_{\mathrm{x}} - 1, v_{\mathrm{y}} + 1, 2^0\right)$ **into** $\mathfrak{L}^A$
  end if
  // Push tuples to sorted queue $\mathfrak{L}^B$
  if $v_{\mathrm{x}} \geq \mathrm{L}$ **and** $v_{\mathrm{x}} \leq \mathrm{R}$ **and** $v_{\mathrm{y}} + 1 \geq \mathrm{T}$ **and** $v_{\mathrm{y}} + 1 \leq \mathrm{B}$ **then**
    PUSH triplet $\left(v_{\mathrm{x}}, v_{\mathrm{y}} + 1, 2^1\right)$ **into** $\mathfrak{L}^B$
  end if
end for
MERGE $\mathfrak{L}^A$, $\mathfrak{L}^B$ **into** $\mathfrak{L}$, **following raster-scan order.**

Algorithm 3 The function **FASTLIMITS** calculates the upper and lower cumulative probability limits of a run and encodes it via an arithmetic encoder. *Inputs* $(v_{\text{sx}}, v_{\text{sy}})$ and $(v_{\text{ex}}, v_{\text{ey}})$ are the x-y coordinates of first and last voxel of the run length whereas the boolean b indicates if the last voxel is occupied or not. Using $C^{\square}, C^{\blacksquare}, \mathcal{C}$ and the queue $\mathfrak{L}$ from Algorithm 1 we allow a fast calculation of the probability limits. In the case the limits would trigger an underflow at the Arithmetic Encoder, we signal the use of an alternative method through appending 1 to the list S^{mod}. *Outupts* are the updated context counts and the flag signaling a fast or sequential-type of run encoding.

INPUTS: $v_{\text{sx}}, v_{\text{sy}}, v_{\text{ex}}, v_{\text{ey}}, b, \mathfrak{L}, C^{\square}, C^{\blacksquare}, \mathcal{C}, S^{\text{mod}}$
OUTPUTS: $C^{\square}, C^{\blacksquare}, \mathcal{C}, S^{\text{mod}}$

$r \leftarrow (v_{\text{ey}} - v_{\text{sy}} - 1) \cdot (\text{R} - \text{L} + 1) + (\text{R} - v_{\text{sx}} + 1) + (v_{\text{ex}} - \text{L})$
$p^{\blacksquare} \leftarrow 1$
$n^{\blacksquare} \leftarrow 0$
$\mathfrak{P} \leftarrow$ **Empty Queue**
$\mathcal{C}_{\text{old}} \leftarrow \mathcal{C}$
// Look at the first element of the queue returning the tuple of the x and y coordiantes of the first voxel with non empty context with context id c
PEEK $(v_{\text{x}}, v_{\text{y}}, c)$ **from** $\mathfrak{L}$
while $(v_{\text{y}} < v_{\text{ey}}$ **or** $v_{\text{x}} < v_{\text{ex}})$ **and** $v_{\text{y}} \leq v_{\text{ey}}$ **do**
// Probabilities of voxels with non empty contexts. Last voxel in run is excluded.
// Remove first queue tuple
POP $\mathfrak{L}$
$w \leftarrow \dfrac{C_c^{\square}}{C_c^{\blacksquare} + C_c^{\square}}$
PUSH $(v_{\text{x}}, v_{\text{y}}, w)$ **into** $\mathfrak{P}$
$p^{\blacksquare} \leftarrow p^{\blacksquare} \cdot w$
$C_c^{\square} \leftarrow C_c^{\square} + 1$
$n^{\blacksquare} \leftarrow n^{\blacksquare} + 1$
PEEK $(v_{\text{x}}, v_{\text{y}}, c)$ **from** $\mathfrak{L}$
end while
// Probabilities of voxels with empty contexts. Last voxel in the run is excluded.
$p^{\square} \leftarrow 1$
if $r > n^{\blacksquare}$ **then**
// If there are any empty context voxels
$p^{\square} \leftarrow \mathcal{C}/(\mathcal{C} + r - n^{\blacksquare})$
$\mathcal{C} \leftarrow \mathcal{C} + r - n^{\blacksquare}$
end if
// Calculate probability and update context counts for last voxel in the run.
PEEK $(v_{\text{x}}, v_{\text{y}}, c)$ **from** $\mathfrak{L}$
if $v_{\text{x}} = v_{\text{ex}}$ **and** $v_{\text{y}} = v_{\text{ey}}$ **then**
// If the final voxel corresponds to a non empty context.
POP $\mathfrak{L}$
$p_{\text{end}} \leftarrow \dfrac{C_c^{\square}}{C_c^{\blacksquare} + C_c^{\square}}$
PUSH $(v_{\text{ex}}, v_{\text{ey}}, p_{\text{end}})$ **into** $\mathfrak{P}$
if b **then**
// if last voxel is occupied
$C_c^{\blacksquare} \leftarrow C_c^{\blacksquare} + 1$
else
// if last voxel is not occupied
$C_c^{\square} \leftarrow C_c^{\square} + 1$
end if
else
$p_{\text{end}} \leftarrow \dfrac{\mathcal{C}}{\mathcal{C} + 1}$
if b **then**
// if last voxel is occupied
$\mathcal{C} \leftarrow \lfloor \mathcal{C}/2 \rfloor$
else
// if last voxel is not occupied
$\mathcal{C} \leftarrow \mathcal{C} + 1$
end if
end if
$p_{\text{up}} \leftarrow p^{\square} \cdot p^{\blacksquare} \cdot p_{\text{end}}$
$p_{\text{low}} \leftarrow p^{\square} \cdot p^{\blacksquare}$
// Calculate upper and lower probability of limits.
if b **then**
// Last voxel was occupied
$p_{\text{up}} \leftarrow 1 - p_{\text{up}}$
$p_{\text{low}} \leftarrow 1 - p_{\text{low}}$
else
$p_{\text{low}} \leftarrow 1 - p_{\text{up}}$
$p_{\text{up}} \leftarrow 1$
end if
// Check if difference of probability pair is lower than the threshold Q, therefore possibly causing undeflow error
if $p_{\text{up}} - p_{\text{low}} < \text{Q}$ **or** $p^{\square} \cdot p^{\blacksquare} \cdot p_{\text{end}} < \text{Q}$ **then**
// Underflow: single mode encoding triggered.
$S \leftarrow$ **SINGLEMODE**$(\mathfrak{P}, \mathcal{C}_{\text{old}}, b, r + 1, v_{\text{sx}}, v_{\text{sy}})$
APPEND S **into** S^{mod}
else
APPEND 0 **into** S^{mod}
ARITHMETIC ENCODER (p_{up}, p_{low})
end if

Algorithm 4 The function **SINGLEMODE**, computes the upper and lower probability limits by partitioning the initial run into a series of sub-runs whose probability pairs do not cause underflow errors. Each sub-run is encoded through an arithmetic encoder. *Inputs* $(v_{\mathrm{sx}}, v_{\mathrm{sy}})$ are the run's x-y coordinates of first voxel, b the occupancy boolean describing the state of the run's last voxel, r the length of the run and $\mathfrak{P}$ a queue composed by triplets containing the x-y coordinates of voxels along with their non-occupancy conditional probability linked to non-empty contexts. *Outputs* list S signaling sub-runs. The probability difference threshold is Q

INPUTS: $\mathfrak{P}, \mathcal{C}, b, r, v_{\mathrm{sx}}, v_{\mathrm{sy}}$
OUTPUTS: S
$n^{\square} \leftarrow 0$
$p_{\mathrm{h}} \leftarrow 0$
$p_{\mathrm{l}} \leftarrow 0$
$p^{\square} \leftarrow 1$
$p^{\blacksquare} \leftarrow 1$
while $r \geq 0$ **do**
  PEEK $(v_{\mathrm{x}}, v_{\mathrm{y}}, p_{\mathrm{t}})$ **from** $\mathfrak{P}$
  // Check if voxel linked to non-empty context
  if $v_{\mathrm{sx}} = v_{\mathrm{x}}$ **and** $v_{\mathrm{sy}} = v_{\mathrm{y}}$ **then**
    $p^{\blacksquare} \leftarrow p^{\blacksquare} \cdot p_{\mathrm{t}}$
    $c \leftarrow 1$
  else
    $n^{\square} \leftarrow n^{\square} + 1$
    $p^{\square} \leftarrow \mathcal{C}/(\mathcal{C} + n^{\square})$
    $c \leftarrow 0$
  end if
  $r \leftarrow r - 1$
  // Find coordinates of next voxel
  if $v_{\mathrm{sx}} + 1 >$ R **then**
    $v_{\mathrm{sx}} \leftarrow$ L
    $v_{\mathrm{sy}} \leftarrow v_{\mathrm{sy}} + 1$
  else
    $v_{\mathrm{sx}} \leftarrow v_{\mathrm{sx}} + 1$
  end if
  $p_{\mathrm{l}} \leftarrow p_{\mathrm{h}}$
  $p_{\mathrm{h}} \leftarrow 1 - p^{\square} \cdot p^{\blacksquare}$
  // Check if difference of probability pair is lower than the threshold Q, therefore possibly causing undeflow error
  if $p_{\mathrm{h}} - p_{\mathrm{l}} <$ Q **or** $1 - p_{\mathrm{h}} <$ Q **then**
    // Termination of sub-run
    // p_{l} is low cumulative probability
    // 1 is high cumulative probability
    ARITHMETIC ENCODER (1, p_{l})
    APPEND 1 **into** S
    // Initiate new of sub-run
    if $c = 0$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} + n^{\square} - 1$
      $n^{\square} \leftarrow 1$
      $p^{\square} \leftarrow \mathcal{C}/(\mathcal{C} + n^{\square})$
      $p^{\blacksquare} \leftarrow 1$
    else
      $\mathcal{C} \leftarrow \mathcal{C} + n^{\square}$
      $n^{\square} \leftarrow 0$
      $p^{\square} \leftarrow 1$
      $p^{\blacksquare} \leftarrow p_{\mathrm{t}}$
    end if
    $p_{\mathrm{l}} \leftarrow 0$
    $p_{\mathrm{h}} \leftarrow 1 - p^{\square} \cdot p^{\blacksquare}$
  end if
end while
if $b = 1$ **then**
    // p_{l} is low cumulative probability
    // p_{h} is high cumulative probability
  ARITHMETIC ENCODER (p_{h}, p_{l})
else
    // p_{h} is low cumulative probability
    // 1 is high cumulative probability
  ARITHMETIC ENCODER (1, p_{h})
end if

APPENDIX A
PROOF OF PROBABILITY LIMITS EQUIVALENCE

Eq. 3 of the Journal states that the probability limits of a run of length l, composed by voxels with contexts c_i are computed by

$$\text{Low}(l) = 1 - \prod_{i=1}^{l-1} p(0|c_i),$$
$$\text{High}(l) = 1 - \prod_{i=1}^{l} p(0|c_i).$$

The proof of Eq. 3 is as follows. The probability of a run of length r, composed by the set of voxels $\{v_1, v_2, ..., v_l\}$ with contexts $\{c_1, c_2, ..., c_l\}$ is given by: $P(r) = p(1|c_r) \cdot \prod_{i=1}^{r-1} p(0|c_i)$.

Therefore the high cumulative probability limit is given by

$$\begin{aligned}
\text{High}(l) &= \sum_{r=1}^{l} P(r) \\
&= p(1|c_1) + \sum_{r=2}^{l} p(1|c_r) \cdot \prod_{i=1}^{r-1} p(0|c_i) \\
&= (1 - p(0|c_1)) + \sum_{r=2}^{l} (1 - p(0|c_r)) \cdot \prod_{i=1}^{r-1} p(0|c_i) \\
&= 1 - p(0|c_1) + \sum_{r=2}^{l} \prod_{k=1}^{r-1} p(0|c_k) - \sum_{r=2}^{l} \prod_{i=1}^{r} p(0|c_i) \\
&= 1 - p(0|c_1) + \sum_{r=1}^{l-1} \prod_{k=1}^{r} p(0|c_k) - \sum_{r=2}^{l} \prod_{i=1}^{r} p(0|c_i) \\
&= 1 + \sum_{r=2}^{l-1} \prod_{k=1}^{r} p(0|c_k) - \sum_{r=2}^{l-1} \prod_{i=1}^{r} p(0|c_i) - \prod_{i=1}^{l} p(0|c_i) \\
&= 1 - \prod_{i=1}^{l} p(0|c_i).
\end{aligned}$$

Idem for the formula of $\text{Low}(l)$.

APPENDIX B
PROOF OF FAST POWER OF PROBABILITY CALCULATION

Described in this section is the proof of Eq. 4 of the Journal, which states that

$$\text{Low}(l) = 1 - \frac{\mathcal{C}}{\mathcal{C} + m} \cdot \prod_{i=1}^{k} p(0|c_i).$$

The starting point of the proof is the following Eq. 1 which separates the terms of voxels associated to the empty contexts $c = 0$ with the non-empty ones $c_i \neq 0$:

$$\text{Low}(l) = 1 - \prod_{i=1}^{m} p(0|c = 0) \cdot \prod_{\substack{i=1 \\ c_i \neq 0}}^{k} p(0|c_i) \qquad (1)$$

We know that all the contexts in the first product term of Eq. 1 are non-occupied. Given that the number of occupied and non-occupied voxels, associated with the empty context are 1 and N respectively, the probaibility of the first non-occupied voxel of context $c = 0$ is $p_1 = \frac{N}{N+1}$. The probability of the second non-occupied voxel with context $c = 0$ is $p_1 = \frac{N+1}{N+1+1}$ since the number N, of non-occupied voxels related to the empty context has been increased by one. Therefore the value N, keeps updaating at each occurrence of a non-occupied voxel of context $c = 0$. Therefore the product of Eq. 1 is rewritten as

$$\begin{aligned}
\prod_{i=1}^{m} p(0|c = 0) &= \prod_{i=1}^{m} \frac{N + i - 1}{N + i} = \frac{\prod_{i=1}^{m} N + i - 1}{\prod_{i=1}^{m} N + i} \\
&= \frac{\prod_{i=1}^{m} N + i - 1}{\prod_{i=2}^{m+1} N + i - 1} \\
&= \frac{(N + 1 - 1) \cdot \prod_{i=2}^{m} N + i - 1}{(N + m + 1 - 1) \cdot \prod_{i=2}^{m} N + i - 1} \\
&= \frac{N}{N + m}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\text{Low}(l) &= 1 - \prod_{i=1}^{m} p(0|c = 0) \cdot \prod_{i=1}^{k} p(0|c_i) \\
&= 1 - \frac{N}{N + m} \cdot \prod_{i=1}^{k} p(0|c_i).
\end{aligned}$$

Chapter 6

Analysis

6.1 Analysis of results

In this section, we discuss the results and the conclusions of the experiments detailed in chapters 2 to 5. First, we touch upon the results related to the attribute compression of point-based data, which is accomplished through graph filterbank transforms. The discussion on this topic is split into two subsections, depending on the way that the graph structure is made available to the decoder. On one hand, the graph structure can be inferred by the decoder without requiring the transmission of additional dedicated graph information, as is the case of hyperspectral image compression through lossy-to-lossless graph filterbank transforms. On the other hand, the graph structure is transmitted as side information when such a graph inference is not possible, as is the case of lossy compression of natural images. After discussing our results on attribute compression, we focus on the topic of compression of point-based data location information. This was accomplished through the geometry compression of point cloud data.

6.1.1 Point-based attribute compression with graph inference at decoder

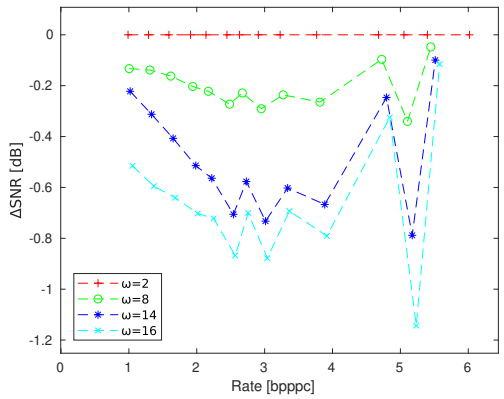
In particular cases of point-based attribute compression applications, the nature of data is such that one can avoid transmitting the graph structure to the decoder when using graph fil-

terbank transforms. One such case is the compression of hyperspectral images, discussed in chapter 2, where we provide different lossy-to-lossless graph filterbank transforms, applied on the spatial dimension of the image. Such proposed transforms are the integer GraphBior (IGB) and the quad or dual kernel integer-to-integer spectral graph lifting (coined ISGL_Q and ISGL_D respectively). To avoid the transmission of the graph through side information, the hyperspectral bands are grouped into packages, called band groups, consisting of ω consecutive bands. Each band group is compressed by a graph transform computed from a single graph, which is derived from the last component of the previously decompressed band group. This circumvents the necessity for the transmission of the actual graph structure, as the graph can be inferred by the decoder.

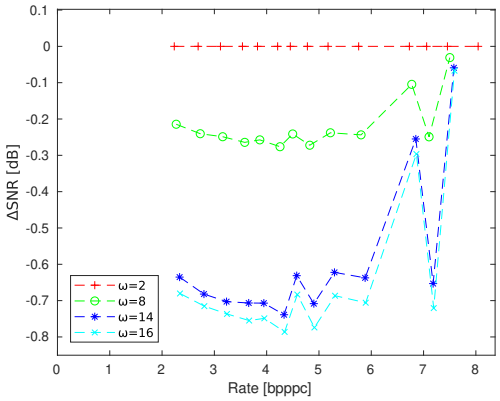
However, the compression of hyperspectral images through graph transforms, based on the inference of the graph in the decoder, has its own difficulties. Of particular concern is the size of ω, dictating the number of components in each band group. Our experimental results displayed in Tale 6.1 and Figures 6.1, show that in the absence of any spectral transform, a smaller band group provides the best results in the lossless as well as lossy cases. This is justified as the correlation of hyperspectral bands is lower the further apart they are located on the spectral dimension. It should be noted though, that the performance of the compression scheme might be hindered by the quality of the original image, such as the existence of dead bands. This effect is particularly pronounced when using a small band group. The probability of using a dead band to compute a band group's graph becomes higher as ω decreases. This is confirmed by the poor results when using lower ω values for the lossy compression of the Hyperion images (Figures 6.1c and 6.1d), as those contain the highest number of dead bands among the corpus used in the experiments.

Table 6.1: Rates at which IGB achieves a lossless compression for different values of ω. The tiles size is set to 16 by 16. The spectral size of the hyperspectral image should be a multiple of ω. Units are in bpppc.

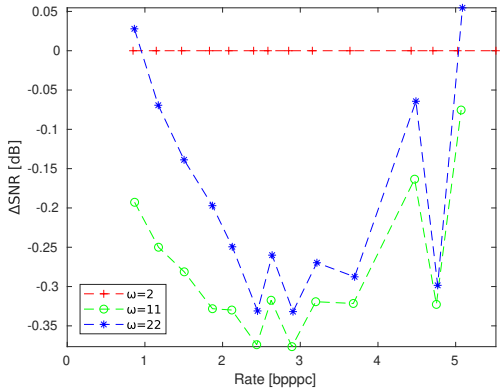
Image	**IGB**						
	$\omega = 2$	$\omega = 3$	$\omega = 8$	$\omega = 11$	$\omega = 14$	$\omega = 16$	$\omega = 22$
Yellowstone sc. 0 cal.	**6.95**		7.00		7.06	7.14	
Yellowstone sc. 0 raw	**9.02**		9.06		9.14	9.15	
Lake Monona	**6.31**			6.36			6.37
Mt. St. Helens	**6.68**			6.75			6.75
Agriculture	**4.27**	4.39					



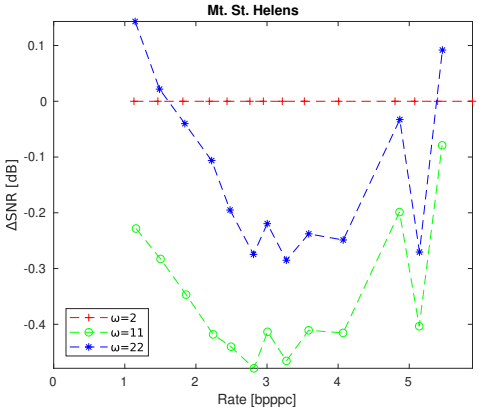
(a) Yellowstone sc. 0 cal



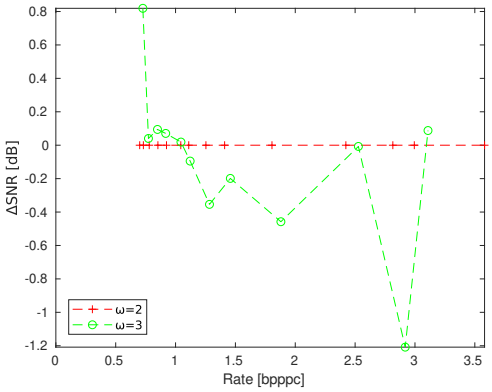
(b) Yellowstone sc. 0 raw



(c) Lake Monona



(d) Mt. St. Helens



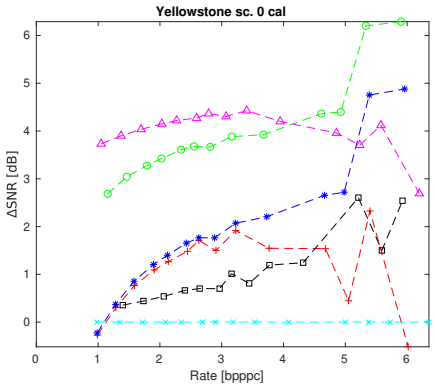
(e) Agriculture

Figure 6.1: Relative rate-distortion plots for IGB using tiles of 16 by 16 when varying the parameter ω for multiple images. Results are relative to those of smallest value of ω.

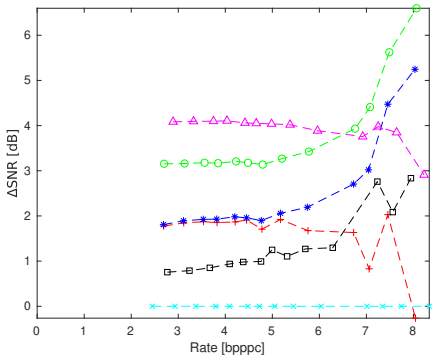
Our comparisons regarding lossy compression, illustrated in Figures 6.2, are performed utilizing transforms only on the spatial dimension of hyperspectral images. These demonstrate that ISGL_Q yields the best overall results, with the exception of compression at high rates, where it is outperformed by the GraphBior transform. Noticeably all the proposed transforms outperform the Discrete Wavelet transform (DWT) in most cases in the lossy setting. Concerning the results on lossless compression, Table 6.2 reveals that the IGB transform most frequently provides the best results, while DWT ranks second in these comparisons.

Table 6.2: Rates at which each integer transform achieves a lossless compression. No spectral transform is used. The parameter ω is set to 2. The IGB transform uses tiles of 16 by 16 for the Yellowstone images and 32 by 32 for Lake Monona, Mt. St. Helens and Agriculture. Units are in bpppc.

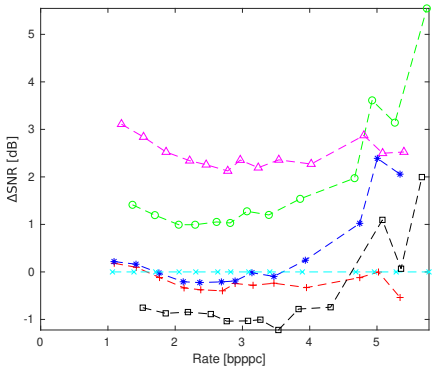
	Transform			
Image	**IGB**	**DWT**	**ISGL_D**	**ISGL_Q**
Yellowstone sc. 0 cal.	**6.95**	7.29	7.44	7.11
Yellowstone sc. 0 raw	**9.02**	9.32	9.47	9.19
Lake Monona	6.26	**6.23**	6.50	6.28
Mt. St. Helens	6.63	**6.57**	6.93	6.67
Agriculture	**4.21**	4.37	4.67	4.37



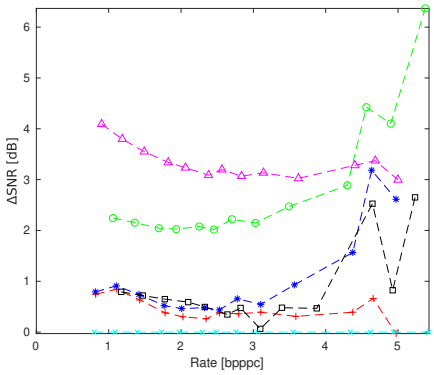
(a) Yellowstone sc. 0 calibrated



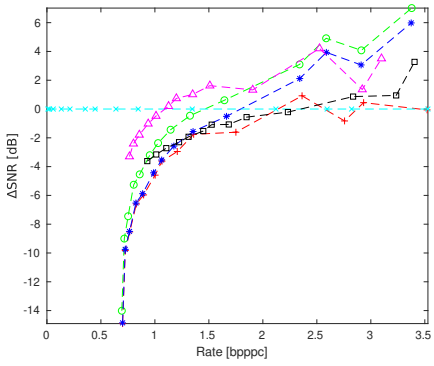
(b) Yellowstone sc. 0 raw



(c) Mt. St. Helens



(d) Lake Monona



(e) Agriculture

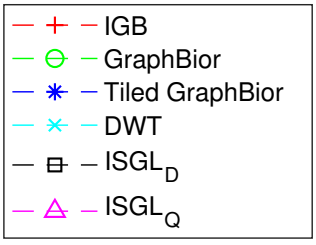
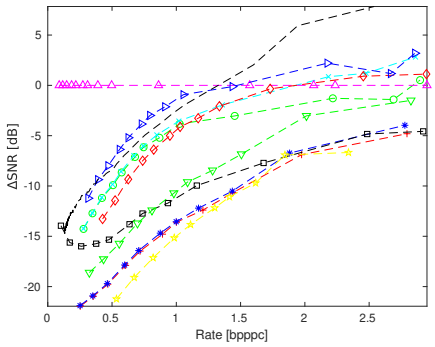


Figure 6.2: Relative rate-distortion plots comparing spatial transforms using $\omega = 2$ for multiple images. Results are relative to DWT. The tiles sizes are 16 by 16 for the Yellowstone sc. 0 images, and 32 by 32 for the rest.

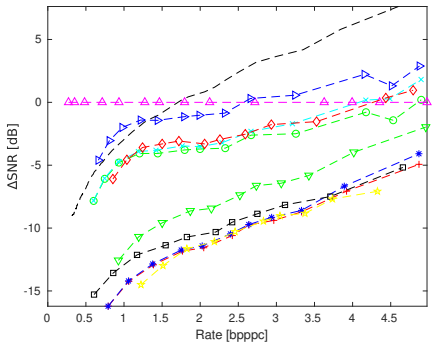
The comparisons between the aforementioned transforms, applied spatially, are further examined by additionally including spectral transforms in the proposed scheme, such as the Reversible Karhunen - Loève Transform (RKLT), and the DWT. Among the transforms used in our compression scheme, the most competitive in the lossy setting, as shown in Figure 6.3, is the ISGL_Q followed by ISGL_D when paired with a spectral RKLT. Both of these transform combinations outperform the spectral RKLT and spatial DWT, mostly in medium and high rates. Regarding the lossless compression of hyperspectral images, Table 6.3 suggests that it is the spatial DWT coupled with the spectral RKLT that provides the most competitive results among the transforms tested in the proposed compression scheme. Nevertheless, when comparing the compression scheme utilizing the proposed graph transforms, against the CCSDS-123.0-B-2 standard [11], Figure 6.3 shows that the latter appears as the most competitive in mid and high rate ranges. At low bitrates it is the spectral RKLT and the spatial DWT or, at a lesser extent, the ISGL_Q, that improve on CCSDS. Although the CCSDS standard consistently outperforms the proposed scheme and transforms in terms of lossless compression, it should be noted that the former is a highly refined method, whereas the latter was devised as a mean to test the viability of graph transforms in the context of hyperspectral image compression.

Table 6.3: Rates at which each of the integer transforms achieves a lossless compression. RKLT and DWT are used as spectral transforms. For the methods that use tiles, their size has been set to 16 by 16. The Yellowstone images are evaluated at $\omega = 8$, the Lake Monona and Mt. St. Helens are evaluated at $\omega = 11$ and the Agriculture image is evaluated at $\omega = 3$. For each column, the spectral transform is mentioned first followed by the spatial transform (spectral transform + spatial transform). Units are in bpppc.

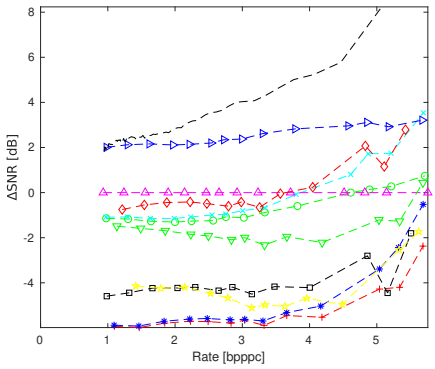
Image	**Transforms**								**CCSDS**
	DWT + IGB	**RKLT + IGB**	**DWT + DWT**	**DWT + ISGL_D**	**DWT + ISGL_Q**	**RKLT + DWT**	**RKLT + ISGL_D**	**RKLT + ISGL_Q**	
Yellowstone sc. 0 cal.	5.03	4.41	4.73	5.36	5.05	3.74	4.65	4.36	4.04
Yellowstone sc. 0 raw	7.17	6.47	6.97	7.54	7.25	5.93	6.72	6.44	6.19
Lake Monona	6.69	6.19	6.56	6.89	6.66	6.35	6.34	6.13	6.10
Mt. St. Helens	7.06	6.52	6.90	7.32	7.05	6.58	6.71	6.47	6.37
Agriculture	4.21	3.96	3.96	4.63	4.34	3.68	4.37	4.08	3.62



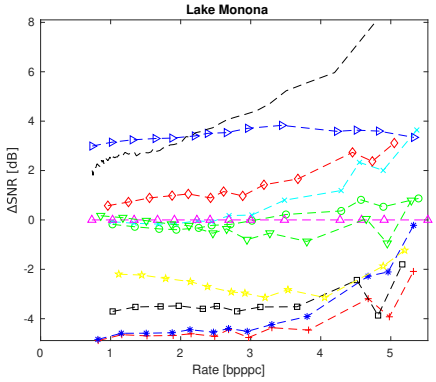
(a) Yellowstone sc. 0 calibrated



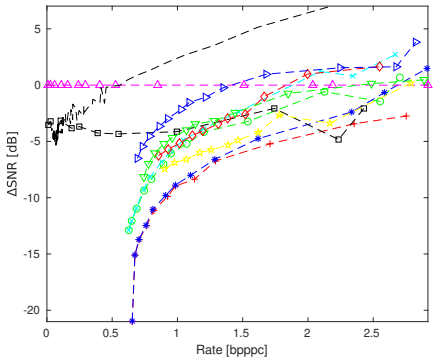
(b) Yellowstone sc. 0 raw



(c) Mt. St. Helens



(d) Lake Monona



(e) Agriculture

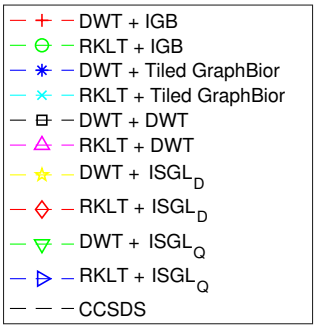


Figure 6.3: Relative rate-distortion plots comparing spectral + spatial transforms for multiple images. Results are relative to the spectral RKLT followed by the spatial DWT. Tiles of 16 by 16 were used. The parameter ω is set to 8 for Yellowstone sc. 0 raw and cal., to 11 for Mt. St. Helens and Lake Monona, and to 3 for Agriculture.

A noticeable drawback of the IGB transform is its high time complexity caused by arduous calculations during the TERM factorization. To rectify this issue, we proposed a tiling method, which segments the hyperspectral components into smaller rectangular tiles. A separate IGB transform is adaptively computed for each individual tile in parallel, therefore reducing the overall time complexity of the process. Our experimental results of Table 6.4 and Figures 6.4 indicate that the smaller the tile size, the lower the compression performance of IGB. This is attributed to the fact that the smaller the tile size, the greater the number of pairs of neighboring pixels belonging to adjacent tiles, whose correlation is therefore never exploited.

Table 6.4: Rates at which integer GraphBior (IGB) achieves a lossless compression using $\omega = 2$ for various tile sizes and multiple images. Units are in bpppc.

Image	**IGB**		
	Tiles of 8 by 8	**Tiles of 16 by 16**	**Tiles of 32 by 32**
Yellowstone sc. 0 cal.	7.14	**6.95**	-
Yellowstone sc. 0 raw	9.15	**9.02**	-
Lake Monona	6.40	6.31	**6.26**
Mt. St. Helens	6.78	6.68	**6.63**
Agriculture	4.39	4.27	**4.21**

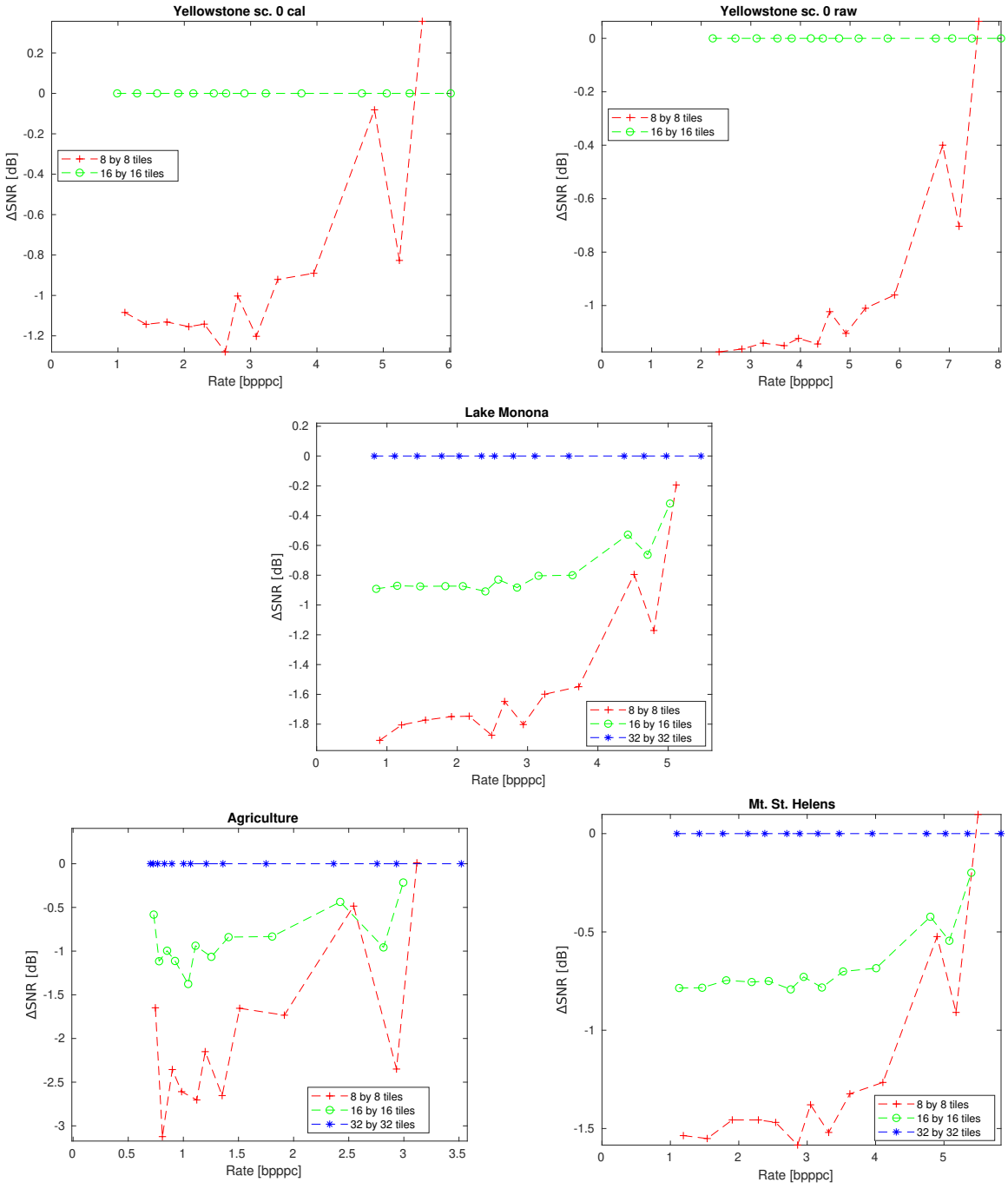


Figure 6.4: Relative rate-distortion plots for IGB, using $\omega = 2$, when varying the tile size, for multiple images. Results are relative to those of the largest tile.

6.1.2 Point-based attribute compression with graph transmission to decoder

For the compression of gray-scale natural images through graph transforms, the aforementioned graph inference scheme is not viable. Instead, it is necessary to transmit the graph structure as side information in order to decode the compressed image. To facilitate the graph transmission, an edge-aware graph is used in such cases, consisting of only two types of weighted edges. The strong edge links neighboring pixels that share a high correlation (e.g. in intensity), while the weak type of edge connects neighboring pixels of lower correlation.

Even though one benefit of edge-aware graphs is their relatively inexpensive transmission cost, they were originally designed to attenuate an undesirable effect linked to the zero-DC GraphBior transform in lossy compression applications. The zero-DC GraphBior, also known as L_{RW}-GraphBior, tends to produce localized artifacts at high gradient regions of the image. The characteristic of L_{RW}-GraphBior, linked to the non-uniform error distribution can be somewhat limited through edge-aware graphs, however, the theoretical foundations, to understand and control this problem, had not been developed at the time. Interestingly, it had been observed that this adverse characteristic was absent in the L_N-GraphBior, when instead of the random walk Laplacian (L_{RW}), the Normalized Laplacian (L_N) would be used as the fundamental matrix. Nonetheless, doing so would result in a significant drop in compression performance, when compared to the L_{RW}-GraphBior variant.

Our work, detailed in chapter 3, shows that the cause of the non-uniform distribution of reconstruction errors is due to L_{RW}-GraphBior analysis and synthesis filters being approximately orthogonal in the degree-inner product. This explains the accumulation of large distortion errors at nodes with a low graph degree value. Therefore, we proposed a novel GraphBior variant based on a novel fundamental matrix, which is designed to balance the trade-off between the high energy compaction and uniform distortion distribution of the L_{RW}-GraphBior and the L_N-GraphBior respectively. This is achieved through an optimization problem, which learns the optimal values of the fundamental matrix according to a learning parameter α. The role of the parameter is to regulate the trade-off between

Image	Experiment	α_1	α_2	min. weight
peppers	1	2.86	$4 \cdot 10^{-4}$	0.1
Barbara	1	4.32	$5.4 \cdot 10^{-3}$	0.23
peppers	2	3.48	$3.48 \cdot 10^{-8}$	10^{-3}

Table 6.5: Table of parameters used in the first and second experiments. Fundamental matrices $\mathbf{Z}_{K1}$and $\mathbf{Z}_{K2}$ are computed through learning parameters α_1 and α_2 respectively. The edge-aware graph uses as a minimum weight the values indicated in the last column.

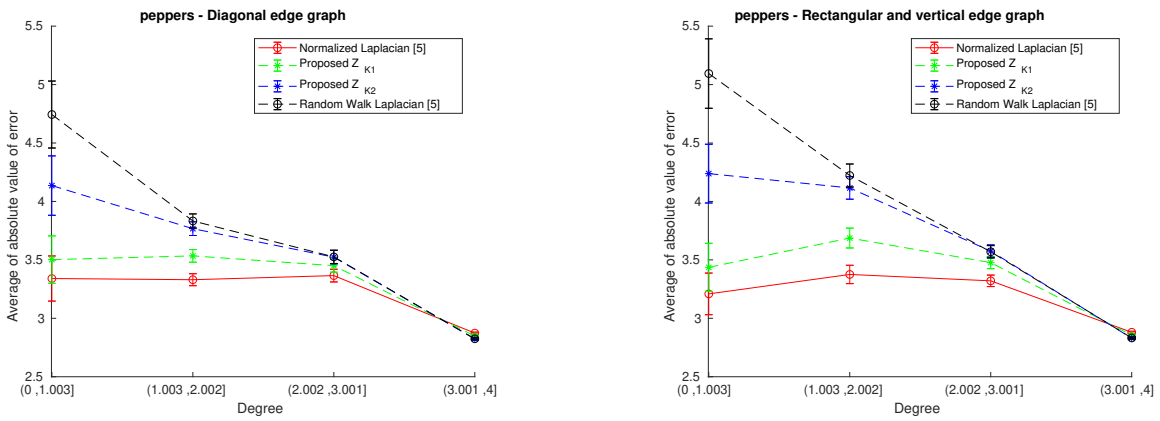
non-uniform error distribution and energy compaction.

Several experiments on the compression of natural images, using the parameters indicated in Table 6.5, confirm that our proposed Z-GraphBior reduces the distortion on high gradient areas of the image, while maintaining a high energy compaction property. Specifically, we examine two different learning parameters α_1 and α_2, resulting in the fundamental matrices Z_{K1} and Z_{K2} respectively. In Table 6.6, we report the results of the first experiment, consisting of the Bjontegaard measurements comparing the Z-, and L_N-GraphBior against the L_{RW}-GraphBior, for the compression of two natural images. Particularly, the performance of each GraphBior variant is measured by computing the distortion on the entire image, as well as on localized high gradient areas of the image, marked as Total and Edge PSNR in Table 6.6 respectively. The results demonstrate an improvement of the proposed Z-GraphBior over L_N-GraphBior in all areas, as well as also outperforming L_{RW}-GraphBior in the PSNR of the high gradient areas of the image. However, the L_{RW}-GraphBior consistently yields the best overall performance, when taking into account the distortion of all the image pixels. Additionally, we were able to showcase the influence of the choice of the learning parameter α. The smaller the parameter, the closer to L_{RW}-GraphBior the proposed GraphBior becomes. In other words, the proposed transform achieves higher energy compaction, while the distortion at high gradient regions of the image increases. Conversely, the larger the value of α the overall compression performance of the method is traded for a more uniform distribution of the image reconstruction errors.

	Peppers				Barbara			
	$\mathbf{L}_N$	$\mathbf{Z}_{K1}$	$\mathbf{Z}_{K2}$	$\mathbf{L}_{RW}$	$\mathbf{L}_N$	$\mathbf{Z}_{K1}$	$\mathbf{Z}_{K2}$	$\mathbf{L}_{RW}$
Total PSNR	-2.005	-0.282	-0.024	**0**	-1.292	-0.132	-0.028	**0**
Edge PSNR	-0.509	**0.614**	0.161	0	-0.332	**0.507**	0.115	0
Rate(%)	36.51	4.854	0.377	**0**	11.317	1.158	0.245	**0**

Table 6.6: Tables of Bjontegaard measurements of experiment 1, comparing the GraphBior variations using $\mathbf{L}_N$ and the proposed $\mathbf{Z}_{K1}$, $\mathbf{Z}_{K2}$ against $\mathbf{L}_{RW}$. The comparisons are done using images *peppers* and *Barbara*

To experimentally demonstrate the relation between the distortion and a pixel's graph degree, in our second experiment, an image is compressed with the L_N-, L_{RW}-, and the proposed Z-GraphBior, such that in all cases the PSNR of the decompressed image is approximately 35.9dB. For each of the bipartite graphs used in GraphBior, all the graph nodes are assembled into four groups, ranging from nodes of low degrees to nodes of high degrees. Then the absolute values of reconstruction errors, of all pixels of each group, are averaged. By plotting in Figure 6.5 the distortion in function of the degree for each of the aforementioned transforms, we demonstrate the tendency towards a uniform error distribution of the L_N-GraphBior and the degree-dependent distortion feature of L_{RW}-GraphBior. Furthermore, we showcased the intermediate behavior of Z-GraphBior, which is dictated by the learning parameter α. A higher parameter will tend to distribute in a more uniform manner the reconstruction errors, resulting in a curve with a flatter profile, colored in green in Figure 6.5.



(a) (b)

Figure 6.5: Experiment 2: Average absolute value of node error with a 95% confidence interval versus degree for (a) Diagonal-edge graph (b) vertical-horizontal-edge graph. The *peppers* image was used and each reconstruction had a total PSNR of approximately 35.9dB

6.1.3 Point-based location compression

Our work on the compression of location information of point based data is studied through the lens of lossless voxelized point cloud geometry compression (PCGC). Despite the emergence of various PCGC methods, the core principle on which most algorithms are based on, is the reduction of non-occupied voxels being processed and encoded. The majority of a point cloud's volume consist of empty space, therefore culminating in a large number of non-occupied voxels. To limit the transmission of empty voxels, the family of octree-based techniques iteratively divides volumes of the point cloud space, that contain at least one occupied voxel, into sub-volumes. Diversely, occupancy matrix techniques such as [12], perform a dyadic decomposition in order to signal through multiple 2D projections, transmitted as side information, areas that are void of any occupied voxel.

The observation that evoked the work detailed in chapter 4 was based on our study of [12] to provide a possible alternative to the dyadic decomposition. We noticed that through the use of 3D contexts, it is more beneficial to reduce the amount of side information transmitted to the decoder for the cost of encoding a larger number of empty voxels. That is because empty voxels are associated with a specific 3D context that renders such voxels inexpensive to encode. Our proposed PCGC method discarded the dyadic decomposition in favor of a single projection of the point cloud along an axis of our choice. The proposed PCGC method was compared against the state-of-the-art inter as well as intra-frame techniques, yielding the best results on a corpus of dynamic point cloud sequences. From Table 6.7 we report average gains higher than 11% over MPEG's G-PCC version 7.0, and above 6% and 2% over the intra and inter-frame dyadic decomposition variants [12, 13] respectively.

Table 6.7: Compression between our proposed methods using the P_z or P_y projection matrix against a plain Octree encoding, G-PCC [14], S-3D [12] and S-4D [13]. We report the average rate in bpov for the lossless geometry compression of the 100 first frames of sequences from the Microsoft Voxelized Upper Bodies and the Full Body JPEG Pleno datasets.

Sequence	Intra Coders					Inter Coder	Gains of **P** over		
	Octree	**G-PCC**	**S-3D**	**Proposed**		**S-4D**	**G-PCC**	**S-3D**	**S-4D**
				Pz	**Py**				
andrew9	2.58	1.14	1.12	**1.02**	1.05	1.08	-10.53%	-8.93%	-5.56%
david9	2.62	1.07	1.06	**0.96**	1.02	1.05	-10.28%	-9.43%	-8.57%
phil9	2.64	1.18	1.14	**1.03**	1.05	1.13	-12.71%	-9.65%	-8.85%
ricardo9	2.59	1.10	1.04	**0.96**	0.97	1.02	-12.73%	-7.69%	-5.88%
sarah9	2.61	1.08	1.07	**0.97**	1.01	1.04	-10.19%	-9.35%	-6.73%
Average	2.61	1.11	1.09	**0.99**	1.02	1.06	-11.29%	-9.01%	-7.12%
longdress	2.99	1.03	0.95	0.94	**0.89**	0.95	-13.59%	-6.32%	-6.32%
loot	2.98	0.97	0.92	0.91	**0.85**	0.91	-12.37%	-7.61%	-6.59%
redandblack	3.00	1.10	1.02	1.00	**0.95**	1.02	-13.64%	-6.86%	-6.86%
soldier	3.00	1.04	0.96	0.93	0.90	**0.81**	-13.46%	-6.25%	11.11%
Average	2.99	1.04	0.96	0.95	**0.90**	0.92	-13.27%	-6.76%	-2.17%

Moreover, we explore the possible benefits of discarding an adaptive arithmetic encoder, in favor of a semi-adaptive one. Our study compares the theoretical average conditional information per occupied voxel, using the true conditional probabilities (marked as LB), as well as the ones computed adaptively (marked as ALB). The reported results of Table 6.8 indicate a large difference between these two theoretical limits, suggesting that the updated probabilities converge slowly to the true ones. Therefore, we provide a simplistic estimation of a semi-adaptive approach, which would transmit as side information (SI) the true conditional probabilities of 4% of the contexts, while the rest of the conditional probabilities would be computed adaptively. By assuming that each of the transmitted probabilities are encoded in a 2-byte word, and each context identifier would require 14 bits, the average estimated gains over our original approach (marked as AR) would cumulate to 6.3%.

Table 6.8: Rates for the compression of the images without taking into account the cost of the projection matrix. We compare the actual rate (AR), its theoretical lower bound using adaptive probabilities (ALB) and its theoretical lower bound using the true probabilities (LB). The rates are averages of the first 100 frames for each sequence and the occupancy array is projected along the z-axis. The average side information for encoding 4% of the contexts is displayed under SI.

Sequence	AR	ALB	LB	SI	Gains of $SI + LB$ over AR
Andrew9	1.01	1.00	0.88	0.07	-5.94%
David9	0.94	0.93	0.83	0.06	-5.32%
Phil9	1.02	1.00	0.90	0.06	-5.88%
Ricardo9	0.94	0.93	0.81	0.08	-5.32%
Sarah9	0.95	0.94	0.82	0.07	-6.32%
Loot10	0.90	0.88	0.82	0.03	-5.56%
Longdress10	0.94	0.92	0.86	0.02	-6.38%
RedAndBlack	0.99	0.98	0.90	0.03	-6.06%
Soldier	0.92	0.90	0.85	0.02	-5.43%

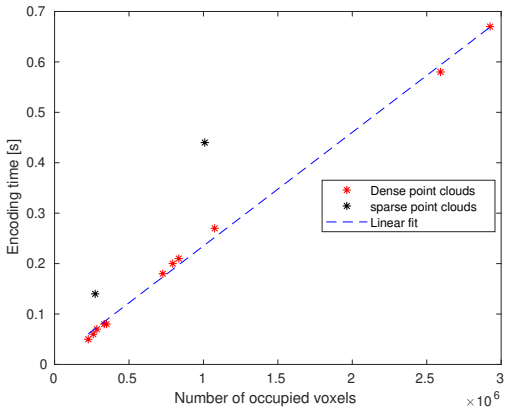
Although our contribution of chapter 4 resulted in a PCGC algorithm of competitive performance, its computational complexity are significantly large. Particularly, its computational complexity is of the order of encoded voxels as each voxel is processed and encoded sequentially. In addition to all the occupied voxels, a large number of empty voxels are also processed, cumulating in slow encoding speeds. In chapter 5 we improve on our previous approach and propose a fast run length point cloud geometry encoder. Experimental results, which are illustrated in Figure 6.6, verify that the proposed fast PCGC encoding algorithm follows a linear complexity with respect to the number of occupied voxels, regardless of the surplus of encoded empty voxels. Furthermore, in Table 6.9, we report average speedups of 1.8 over MPEG's standard TMC13 version 14. In Table 6.10, our comparisons on the performance of our approach against the state of the art, show that we consistently outperform other intra, non-neural network approaches, reporting gains of up to 15% over TMC13. Our proposal even outperforms the MSVDNN technique [15], based on neural networks on the upper body point cloud dataset. Furthermore, although the neural network-based methods [15, 16] usually outperform the proposed method in terms of rate, the former's high computational complexity and slow encoding speeds render the latter the preferred method.

Table 6.9: Encoding time comparison between proposed method *FRL* and state of the art. Measurements are averaged on all frames of each sequence and displayed in seconds.

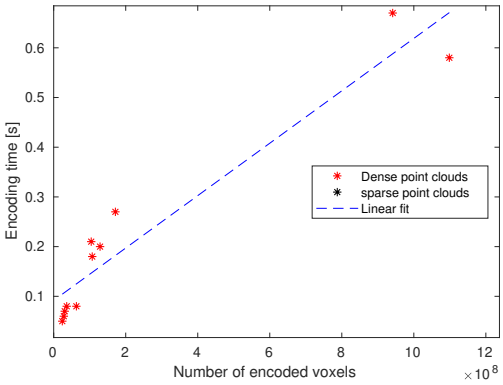
Sequence	**FRL**			**TMC13 [17]**	**Speedup over TMC13**
	Sorting	Encoding	Total		
andrew9	0.005	0.07	**0.07**	0.10	1.4
david9	0.006	0.08	**0.09**	0.13	1.4
phil9	0.006	0.08	**0.08**	0.12	1.5
ricardo9	0.004	0.05	**0.06**	0.08	1.3
sarah9	0.005	0.06	**0.06**	0.09	1.5
Average	0.005	0.07	**0.07**	0.10	1.4
longdress	0.017	0.21	**0.22**	0.31	1.4
loot	0.016	0.20	**0.21**	0.29	1.4
redandblack	0.014	0.18	**0.20**	0.28	1.4
soldier	0.021	0.27	**0.29**	0.40	1.4
Average	0.017	0.22	**0.23**	0.32	1.4
basketball	0.061	0.67	**0.74**	1.05	1.4
dancer	0.054	0.58	**0.64**	0.92	1.4
Average	0.058	0.63	**0.69**	0.99	1.4
Egyptianmask	0.009	0.14	**0.15**	0.48	3.2
Shiva	0.025	0.44	**0.47**	1.30	2.8
Average	0.017	0.29	**0.31**	0.89	2.9

Table 6.10: Rate comparison between proposed method *FRL* and state of the art. Measurements are averaged over all as well as the 100 first frames of each sequence and displayed in bits per occupied voxel [bpov]

Sequence	**All frames** (Intra coders)				**Unspecified number of frames** (Intra coders)		**First 100 frames** (Intra coder)	(Inter coder)
	FRL	**SP [18]**	**S3D [12]**	**TMC13 [17]**	**VDNN [16]**	**MSVDNN [15]**	**FRL**	**S4D [19]**
andrew9	**1.00**	1.02	1.12	1.13	-	-	1.01	**0.95**
david9	**0.94**	0.96	1.05	1.07	-	-	**0.94**	**0.94**
phil9	1.02	1.04	1.14	1.17	**0.92**	-	**1.02**	**1.02**
phil10	0.95	-	-	-	**0.83**	1.02	0.95	-
ricardo9	0.93	0.95	1.03	1.07	**0.72**	-	0.94	**0.90**
ricardo10	0.89	-	-	-	**0.75**	0.95	0.90	-
sarah9	**0.95**	0.97	1.06	1.07	-	-	0.96	**0.92**
Average	0.95	0.99	1.08	1.10	**0.81**	0.99	0.96	**0.95**
longdress	**0.86**	0.89	0.95	1.02	-	-	**0.86**	0.88
loot	0.83	0.86	0.92	0.97	0.64	**0.63**	**0.82**	0.84
redandblack	0.94	0.96	1.03	1.09	**0.73**	0.87	**0.93**	0.94
soldier	**0.88**	0.91	0.97	1.04	-	-	0.88	**0.65**
Average	0.88	0.91	0.97	1.03	**0.69**	0.75	0.87	**0.83**
bask/player	**0.80**	-	-	0.90	-	-	-	-
dancer	**0.77**	-	-	0.89	-	-	-	-
Average	**0.79**	-	-	0.90	-	-	-	-
Egyp/mask	18.20	-	-	**11.78**	-	-	-	-
Shiva	15.11	-	-	**9.68**	-	-	-	-
Average	16.66	-	-	**10.73**	-	-	-	-



(a) Occupied voxels vs encoding time (average)



(b) Encoded voxels vs encoding time (average). Sparse data points are out of view with an encoding time of sub 0.44s and a number of encoded voxels of $2.9 \cdot 10^{10}$ and $5.6 \cdot 10^{10}$

Figure 6.6: Observed relation between encoding time and voxel count.

Finally, in Table 6.11 we show that in replacing the single projection by the signaling rectangle, the transmitted side information further decreases. Although this results in transmitting a significant surplus of empty voxels, this does not impact the overall performance in compression results nor in encoding speeds. In fact, we are able to report a slight gain in compression performance over our single projection approach of chapter 4.

Table 6.11: Coding cost of projection matrix versus signaling rectangle. The right-most column informs on the percentage surplus of encoded voxels if the signaling rectangle is used instead of the projection matrix. Measurements are averaged over all frames of each sequence.

Sequence	**Projection matrix (bpov)**	**Signaling rectangle (bpov)**	**Surplus of encoded voxels (%)**
andrew9	0.0170	$22.5 \cdot 10^{-5}$	145.15
david9	0.0188	$18.3 \cdot 10^{-5}$	178.19
phil9	0.0221	$19.2 \cdot 10^{-5}$	80.31
ricardo9	0.0168	$28.3 \cdot 10^{-5}$	84.29
sarah9	0.0186	$24.6 \cdot 10^{-5}$	120.32
Average	0.0187	$22.6 \cdot 10^{-5}$	121.65
longdress	0.0031	$7.7 \cdot 10^{-5}$	75.52
loot	0.0049	$8.1 \cdot 10^{-5}$	89.50
redandblack	0.0060	$8.8 \cdot 10^{-5}$	89.68
soldier	0.0034	$6.0 \cdot 10^{-5}$	92.05
Average	0.0044	$7.7 \cdot 10^{-5}$	86.69

Chapter 7

Conclusions

7.1 Conclusion

Our contribution to the field of point-based data compression has been summarized in the aforementioned four publications, detailed in chapters 2 to 5. Our research has covered both aspects of point-based data encoding, exploring novel methods for the compression of a point's attribute as well as location information. The former was accomplished through research of graph filterbanks for the compression of natural and hyperspectal images, whereas the latter was achieved through our work on point cloud geometry compression. As a result, the conclusions of this dissertation are organized into three subsections, starting with a few words relevant to our research on graph filterbanks, and later on, on the topic of point cloud geometry compression. Our conclusions then end on some general thoughts, providing a wider scope in the synopsis of the aforementioned research areas.

7.1.1 Conclusions on spectral graph filterbanks

An image can be represented as a graph by considering each pixel as a node, while inter-pixel correlations are translated by connecting them via weighted graph edges. The structure of a graph apart from topological information, such as which nodes share common edges, also includes the value of each weighted edge. Weighted edges tend to be large when connecting nodes of similar values, for instance in smooth areas of an image, whereas they

are lower at areas where node values are less correlated, e.g. high image gradient regions. These are the regions where one encounters sharp intensity changes within pixels of a small neighborhood. The graph structure is of paramount importance, since graph transforms are fully dependent on the graph. Even removing just a few weighted links of a well designed graph can significantly affect the performance of the transform. Such an effect was observed during our research on an integer-to-integer GraphBior variant for hyperspectral image compression, calculated through a triangular elementary matrix (TERM) factorization. To reduce the process's slow execution time, a tiling strategy was adopted, which partitioned the graph into smaller, rectangular grid graphs of specific size. This highly parallelized process greatly increased the factorization speed. However, the smaller the tile size, the poorer was the compression performance. This was accredited to the fact of severing a large amount of cross-tile edges during the partitioning process, strongly affecting the algorithm's performance. This led us to the development of a faster integer-to-integer GraphBior variant utilizing spectral graph lifting structures.

Noticed throughout our research, an important drawback of graph based transforms, in the context of image compression, is the necessity of transmitting the graph structure to the decoder. The means of resolving this issue is highly dependent on the nature of the data, while each strategy is met with its own challenges. In the case of hyperspectral images the graph structure can be made available to the decoder from previously decompressed components. Particularly, the hyperspectral components are grouped into bundles called bandgroups. Those are later compressed sequentially utilizing for each group a discrete graph, which is computed from the last component of the previously compressed group. Although this method of sharing the graph's structural information to the decoder does not require any side information, it presents its own challenges. Parameters such as the size of each bandgroup, ω, affect significantly the performance of the compression scheme and should be selected carefully. Our research shows that when applying graph filterbank transforms on the spatial dimensions of the image, and in the absence of any transforms along the spectral dimension, it is preferable to utilize smaller values of ω. However, this comes in contrast to the case of additionally incorporating a transform on the spectral dimension. As the underlying characteristic of hyperspectral images is their high correlation along the spectral dimension, the latter scenario favors bandgroups of larger size even though this

acts as a detriment to spatially applied graph filterbanks.

This strategy however does not apply to natural images and therefore it is necessary to transmit, as side information, the graph structure to the decoder. Typically, even the sparser graphs that are employed in image compression applications utilize a large number of weighted edges that can easily amount to four times the number of pixels. In order to decrease the amount of side information related to the graph structure, an edge-aware design is adopted. The edge-aware graph is inexpensively transmitted, since its structure embeds less information. This is achieved by utilizing only two types of weighted edges, one to express strong links between nodes and another that expresses weak bonds. Research has shown that such graphs impact the behavior of graph transforms, such as the zero-DC GraphBior variant that is preferred for the compression of natural images due to its high energy compaction quality. It is known that edge-aware graph designs were constructed to benefit the compression performance of the above GraphBior variant, by restricting filtering across boundaries. Particularly, it had been noticed that the zero-DC GraphBior accumulated large reconstruction errors at high gradient areas of the image. Hence, the edge-aware graphs were meant to limit undesirable artifacts that can hinder the appearance of the reconstructed image. However, there was no clear solution on controlling the extent of such artifacts as the cause of the phenomenon was still unknown. Our work identifies the root of this problem, which is caused by the type of inner product in which the GraphBior variant is biorthogonal to. To rectify this drawback, we propose a method that balances the trade off between energy compaction and uniform reconstruction error distribution through a novel Graphbior variant termed the Z-GraphBior.

7.1.2 Conclusions on point cloud geometry compression

Touching upon the subject of the compression of point-based data location information, graph transforms do not appear as a popular choice. Limitations in the acquisition techniques of geometric data, such as the geometry information of point clouds, render the underlying graph signal noisy and irregular. Furthermore, the difficulties of transmitting the necessary information for the graph construction are far more pronounced in this case. Even so, graph based methods, devised for lossy PCGC, are found in the literature. For ex-

ample, [20] utilizes graphs constructed from a previously decoded frame of a time varying point cloud sequence in order to compress future frames.

Therefore, alternative methods are devised to losslessly compress point cloud geometry. As there is an increase in point-cloud-oriented applications, the subject of point cloud geometry compression is a rapidly evolving field. Most PCGC algorithms are based on the octree decomposition, which is known for its relative simplicity and high execution speed, while recent occupancy-array-based approaches demonstrate significant improvements in compression ratio. In both cases though, contexts play a very important role, allowing the encoder to predict the occupancy status of a voxel. To predict if a voxel is occupied or not, each context takes into account the occupancy status of a number of its neighboring voxels. The number of voxels taken into consideration depends on the size of the context, but the larger the context size, the larger the set of possible contexts.

The majority of occupancy-array type of methods employ various schemes to restrict the amount of encoded empty voxels. This is achieved by transmitting to the decoder the boundaries outside of which, no occupied voxels are present. As the precision and accuracy of the boundary information increases, fewer non-occupied voxels are encoded at the cost of a larger amount of side information. Through our work, we show that the cost of such highly accurate side information greatly surpasses the cost of encoding additional non-occupied voxels. This appears to be the case given that the majority of additionally encoded non-occupied voxels is linked to a very specific context that is composed entirely by non-occupied voxels. Therefore, this results in the occupancy of such voxels being highly predictable, rendering them extremely inexpensive to encode. However, the large number of non-occupied voxels being processed and transmitted might influence negatively the speed of the encoder. To rectify this problem, we have found that it is preferable to encode runs of voxels instead of processing them individually.

7.1.3 General conclusions

Taking a step back from our published work, it is worth discussing some general conclusions, as well as predictions concerning the future direction of the topic of point-based data compression. It appears that compression schemes that apply graph filterbanks on natural

images do not yield a significant advantage compared to classical approaches. This deficiency is likely caused by the lack of sharp edges in natural images coupled with the strict weight constraints enforced by the edge-aware graphs. Additionally the necessity to transmit the graph structure as side information further detriments the performance of such an approach. Contrarily, edge-aware graphs coupled with graph filterbanks have reported very promising results for the compression of image data with sharp image gradients such as depth images [21].

Graph filterbanks are also linked to strong compression performances when the graph structure is not required to be transmitted to the decoder. In that case the edge-aware graph is replaced in favor of a more complex graph design, i.e. weighted edges are computed via the bilateral filter [22], such as in the scenario of hyperspectral image compression [23]. Concerning spatial hyperspectral transforms, graph filterbanks, such as the ones based on GraphBior, have been shown to outperform in the lossy, as well as in the lossless domain classical filterbank transforms such as the DWT. However, classical decorrelation transforms applied on the spectral dimension of hyperspectral images, such as the Karhunen - Loève transform, vastly outperform spectrally applied graph filterbanks.

Regarding the compression performance of lossless voxelized PCGC algorithms, it is important to highlight the significance of contexts, as they play a fundamental role in the development of novel methods. The reason why the occupancy-array-based methods seem to outperform the octree approach is due to the former providing the necessary framework for introducing novel three dimensional contexts. By providing the means to render the entirety of the voxel space, occupancy-array methods enable the deployment of highly malleable three dimensional contexts that predict very well the occupancy status of a voxel. Furthermore, it is well supported that 3D contexts are much more competitive than their 2D counterparts. During the occupancy prediction of a given voxel, the 2D contexts consider only the occupancy status of neighboring voxels lying on the same two dimensional plane. Conversely, the three dimensional contexts consider the status of neighboring voxels in an entire sub-volume of the voxel space.

Although 2D contexts are typically composed by fewer voxels than its 3D counterpart, the performance of PCGC algorithms is not absolutely dependent on the context's size. At first glance, it would seem that the larger the context, the better its predictive power, given

that it is able to capture more information from the target voxel's neighborhood. This in turn should cumulate to better compression results. In practice however, this is not the case due to the curse of dimensionality coupled with the fact that adaptive entropy encoders are the standard in PCGC. In other words, the larger the context size, the slower the adaptive encoder will learn the underlying conditional distribution of the point cloud, therefore greatly hindering the compression performance. One way to bypass this issue is by discarding the adaptive encoder for one employing a pre-trained model. This approach has been taken by the neural network (NN) PCGC methods which employ larger 3D contexts but learn beforehand the distribution from a large sample of point cloud data. Although this type of PCGC algorithms are still in their infancy, they have demonstrated a strong compression performance, paving, in my opinion, a new path of research to the topic of interest.

Despite NN approaches vastly outranking the octree-based methods, in terms of compression ratio, this cannot be stated when referring to their computational complexity and therefore execution speed. Neural networks are highly complex techniques requiring powerful GPUs to execute in parallel a huge number of operations. In contrast, octrees are known for their fast execution speed, which constitutes the main reasons why compression standards such as MPEG's lossless PCGC method have adopted this technique. Therefore, it does not seem that NN methods will render approaches, such as octree-based algorithms, redundant in the near future. Furthermore, we might see more hybrid-type methods combining the octree decomposition with classical occupancy array encoding schemes to accelerate compression speeds. Such hybrid approaches have already been employed in neural network PCGC methods, though the network's high complexity appears to be a large bottle neck in terms of execution speed.

7.2 Future Work

Recent developments on the topic of point-based data compression has opened up multiple paths for future investigation. Regarding our research focused on graph filterbank transforms, the area of proper graph construction is still an open problem and a very interesting subject. Specifically, in the case of edge-aware graphs, future research could explore the

trade-off between the transmission of specific edges and transform performance. Furthermore, it is known that edge-aware graph designs that utilize only two weighted edge values are not well suited for the compression of natural images due to the lack of abrupt changes in the image's pixel intensities. Therefore, methods that would embed in the graph a soft transition of pixel intensities by smoothing the weighted edge values of an edge aware-design, could ameliorate the graph filterbank's compression performance.

Additionally, highly interesting is the prospect of developing new 3D filterbank transforms for the compression of hyperspectral images. This could be achieved by designing new filterbanks through the newly introduced hypergraph signal processing [24]. This extension of graph signal processing defines methods to capture high order interactions utilizing a tensor representation. More-so, existing graph filterbank transforms that have been applied on hyperspectral images can be improved through recent developments on the theory of GraphBior [5]. Due to a particular constraint of the original GraphBior framework, the employed graph was required to be bipartite. This would result in graphs connecting nodes with very limited patterns. Such examples would be connecting nodes with their 4 east, west, north and south-most closest neighbors. These limitations are no longer the case, therefore allowing the freedom of choice in defining the graph topology.

Also benefiting from the extension of the GraphBior theory are areas related to attribute compression of point cloud data. The main challenges that earlier approaches faced were related to the construction of a bipartite graph structure from the point cloud geometry [25]. However, it is now possible, given the geometry information of a point cloud, to compose GraphBior filterbanks on an arbitrary topology, whose edges can be weighted according to the inverse of the euclidean distance of connected nodes [5, 26]. This important step could bring to the forefront GraphBior filterbanks for the attribute compression of point clouds, and trigger a plethora of further research on the use of graph filterbanks on such data.

On the subject of point cloud geometry compression, several articles highlight the importance of contexts in occupancy-array based methods, though existing context selection techniques are mostly an untouched subject. The development of optimization problems could be used for a data driven selection of the best suited set of contexts, along with their orientation, for the geometry compression of a target point cloud. This would allow the design of optimal contexts when working under the constraint of context size, which, as

discussed previously, affects the performance of adaptive entropy encoders.

In the previous section, we highlighted the promising results that neural networks-type PCGC methods yield and pointed out that their gains are related to the large size of employed contexts. In order for such methods to by-pass the curse of dimensionality, which heavily burdens adaptive entropy encoders, their distribution estimations are learned from a large set of point cloud data. However, the learned distribution in some cases might be too general and not reflect the particularities of an individual point cloud. Therefore, it might be interesting to explore semi-adaptive approaches, employing a mix of pre-learned context and contexts that are adaptively learned throughout the encoding process.

Bibliography

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[2] S. K. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2786–2799, 2012.

[3] ——, "Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs," *IEEE transactions on signal processing*, vol. 61, no. 19, pp. 4673–4685, 2013.

[4] D. B. Tay, Y. Tanaka, and A. Sakiyama, "Critically sampled graph filter banks with polynomial filters from regular domain filter banks," *Signal Processing*, vol. 131, pp. 66–72, 2017.

[5] E. Pavez, B. Girault, A. Ortega, and P. A. Chou, "Spectral folding and two-channel filter-banks on arbitrary graphs," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5070–5074.

[6] D. C. Garcia and R. L. de Queiroz, "Intra-frame context-based octree coding for point-cloud geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1807–1811.

[7] ——, "Context-based octree coding for point-cloud video," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1412–1416.

[8] R. L. de Queiroz, D. C. Garcia, P. A. Chou, and D. A. Florencio, "Distance-based probability model for octree coding," *IEEE Signal Processing Letters*, vol. 25, no. 6, pp. 739–742, 2018.

[9] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2019.

[10] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph spectral image processing," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 907–930, 2018.

[11] Consultative Committee for Space Data Systems, *Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression. Recommendation for Space Data System Standards, CCSDS 123.0-B-2*. CCSDS, Feb. 2019. [Online]. Available: https://public.ccsds.org/Pubs/123x0b2.pdf

[12] E. Peixoto, "Intra-frame compression of point cloud geometry using dyadic decomposition," *IEEE Signal Processing Letters*, vol. 27, pp. 246–250, 2020.

[13] E. Peixoto, E. Medeiros, and E. Ramalho, "Silhouette 4d: An inter-frame lossless geometry coder of dynamic voxelized point clouds," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 2691–2695.

[14] 3DG, *G-PCC codec description v4*. Tech. Rep., ISO/IEC JTC 1/SC 29/WG 11 input document w18673, 2019.

[15] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Multiscale deep context modeling for lossless point cloud geometry compression," in *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2021, pp. 1–6.

[16] ——, "Learning-based lossless compression of 3d point cloud geometry," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4220–4224.

[17] "C++ MPEG group TMC13 implementation," [Online] Available: https://github.com/MPEGGroup/MPEG-pcc-tmc13, accessed: May 2022.

[18] D. E. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagristà, "Compression of point cloud geometry through a single projection," in *2021 Data Compression Conference (DCC)*. IEEE, 2021, pp. 63–72.

[19] E. Ramalho, E. Peixoto, and J. E. G. de Medeiros, "Silhouette 4d with context selection: Lossless geometry compression of dynamic point clouds," *IEEE Signal Processing Letters*, 2021.

[20] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3d point cloud sequences," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, 2016.

[21] S. K. Narang, Y.-H. Chao, and A. Ortega, "Critically sampled graph-based wavelet transforms for image coding," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 2013, pp. 1–4.

[22] A. Gadde, S. K. Narang, and A. Ortega, "Bilateral filter: Graph spectral interpretation and extensions," in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 1222–1226.

[23] J. Zeng, G. Cheung, Y.-H. Chao, I. Blanes, J. Serra-Sagristà, and A. Ortega, "Hyperspectral image coding using graph wavelets," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1672–1676.

[24] S. Zhang, Z. Ding, and S. Cui, "Introducing hypergraph signal processing: Theoretical foundation and practical applications," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 639–660, 2019.

[25] A. Anis, P. A. Chou, and A. Ortega, "Compression of dynamic 3d point clouds using subdivisional meshes and graph wavelet transforms," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6360–6364.

[26] E. Pavez, B. Girault, A. Ortega, and P. A. Chou, "Two channel filter banks on arbitrary graphs with positive semi definite variation operators," *arXiv preprint arXiv:2203.02858*, 2022.